РУКОВОДСТВО АДМИНИСТРАТОРА LINUX

ВТОРОЕ ИЗДАНИЕ

title.indd 1 04.04.2007 14:26:03

LINUX ADMINISTRATION HANDBOOK

SECOND EDITION

Evi Nemeth
Garth Snyder
Trent R. Hein

with Lynda McGinley, Ben Whaley, Adam Boggs,
Jeffrey S. Haemer, Tobi Oetiker, Fritz Zaucker,
Scott Seidel, Bryan Buus, Ned McClain,
and David Schweikert

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

title.indd 2 04.04.2007 14:26:04

РУКОВОДСТВО АДМИНИСТРАТОРА LINUX

ВТОРОЕ ИЗДАНИЕ

Эви Немет
Гарт Снайдер
Трент Р. Хейн

при участии Линды Мак-Гинли, Бена Вэйли, Адама Боггза, Джефри С. Химера, Тоби Отикера, Фрица Цокера, Скотта Сидела, Брайана Бууса, Неда Мак-Клейна и Девида Швикерта

Издательский дом "ВИЛЬЯМС"
Москва • Санкт-Петербург • Киев
2011

title.indd 3 04.04.2007 14:26:04

ББК 32.973.26-018.2.75
Н50
УДК 681.3.07

Издательский дом “Вильямс”

Зав. редакцией *С.Н. Тригуб*

Перевод с английского *Я.П. Волковой, Д.Я. Иваненко, Ю.И. Корниенко*

Под редакцией *Ю.Н. Артеменко*

По общим вопросам обращайтесь в Издательский дом “Вильямс” по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Немет, Эви, **Снайдер**, Гарт, **Хейн**, Трент.

Н50 Руководство администратора Linux, 2-е издание. : Пер. с англ. — М. : ООО “И.Д. Вильямс”, 2011. — 1072 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-1093-6 (рус.)

Книга общепризнанных “гуру” в области системного администрирования Linux представляет собой всеобъемлющее руководство по всем аспектам инсталляции, развертывания, настройки и сопровождения систем любого уровня сложности на базе широко используемых дистрибутивов этой замечательной операционной системы. Среди рассматриваемых систем — Red Hat Enterprise Linux, Fedora Core, SUSE Linux Enterprise, Debian GNU/Linux и Ubuntu Linux. Круг вопросов простирается от базовых задач администрирования для многопользовательских систем до рассмотрения таких сложных проблем, как организация работы в сетях, доступ в Internet и тонкая настройка производительности. Изобилие ценных советов и проверенных многолетним опытом методик превращает эту книгу в поистине бесценный источник знаний, необходимых в повседневной работе любого системного администратора.

Книга рассчитана на профессионалов в области информационных технологий, однако будет полезной и для новичков, только-только постигающих тонкости столь увлекательной и трудной работы, каковой является системное администрирование.

ББК 32.973.26-018.2.75

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства Prentice Hall, Inc.

Authorized translation from the English language edition published by Prentice Hall, Ptr., Copyright © 2007 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Red Hat Enterprise Linux and the Red Hat SHADOWMAN logo are registered trademarks of Red Hat Inc., and such trademarks are used with permission.

Ubuntu is a registered trademark of Canonical Limited, and is used with permission.

Fedora is a trademark of Red Hat Inc., and is used with permission.

Novell, the Novell logo, the N logo, and SUSE are registered trademarks of Novell Inc. in the United States and other countries.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2011

ISBN 978-5-8459-1093-6 (рус.)
ISBN 0-13-148004-9 (англ.)

© Издательский дом “Вильямс”, 2011
© Pearson Education, Inc., 2007

Оглавление

ЧАСТЬ I. Основы администрирования 39
Глава 1. С чего начать 41
Глава 2. Запуск и останов системы 61
Глава 3. Сила привилегий 85
Глава 4. Управление процессами 95
Глава 5. Файловая система 113
Глава 6. Подключение новых пользователей 137
Глава 7. Добавление жесткого диска 157
Глава 8. Периодические процессы 197
Глава 9. Резервное копирование 205
Глава 10. Система Syslog и журнальные файлы 251
Глава 11. Управление программным обеспечением и конфигурацией 275

ЧАСТЬ II. Работа в сетях 323
Глава 12. Сети TCP/IP 325
Глава 13. Маршрутизация 389
Глава 14. Сетевые аппаратные средства 407
Глава 15. Система доменных имен 431
Глава 16. Сетевая файловая система 543
Глава 17. Совместное использование системных файлов 563
Глава 18. Электронная почта 591
Глава 19. Управление сетями 713
Глава 20. Безопасность 741
Глава 21. Web-хостинг и серверы Internet 793

ЧАСТЬ III. Разное 813
Глава 22. Система X Window System 815
Глава 23. Печать 835
Глава 24. Обслуживание аппаратных средств и условия эксплуатации 869
Глава 25. Анализ производительности 881
Глава 26. Взаимодействие с Windows 899
Глава 27. Устройства последовательного доступа 923
Глава 28. Драйверы и ядро 949
Глава 29. Демоны 967
Глава 30. Стратегии и методики управления и администрирования 987
Предметный указатель 1061

Содержание

Об авторах 31
Предисловие к первому изданию 33
Введение 34
Демонстрационные дистрибутивы 34
Структура книги 35
Наши помощники 36
Информация для контактов 36
Благодарности 37
От издательства 38

ЧАСТЬ I. Основы администрирования 39

Глава 1. С чего начать 41
1.1. Что необходимо знать 42
1.2. Взаимосвязь Linux и UNIX 43
1.3. Linux в историческом контексте 43
1.4. Дистрибутивы Linux 44
Какой дистрибутив наилучший 47
Административные средства конкретных дистрибутивов 47
1.5. Типографские особенности книги 48
Информация по конкретным дистрибутивам 49
1.6. Где искать информацию 49
Организация страниц интерактивного руководства 50
Команда `man`: чтение страниц интерактивного руководства 51
Другие источники информации по Linux 52
1.7. Как инсталлировать программное обеспечение 53
1.8. Основные задачи системного администратора 55
Добавление, удаление и управление учетными записями пользователей 55
Подключение и удаление аппаратных средств 55
Резервное копирование 55
Инсталляция и обновление программ 55
Мониторинг системы 56
Поиск неисправностей 56
Ведение локальной документации 56
Пристальное слежение за системой безопасности 56
Оказание помощи пользователям 57
1.9. Издержки профессии 57
Синдром хронического администрирования 57
1.10. Рекомендуемая литература 58
1.11. Упражнения 59

Глава 2. Запуск и останов системы 61
2.1. Начальная загрузка 61
Автоматическая и ручная загрузка 62
Этапы загрузки 62

Инициализация ядра 63
Конфигурирование аппаратных средств 63
Процессы ядра 63
Действия оператора (только при ручной загрузке) 64
Выполнение сценариев запуска системы 65
Работа в многопользовательском режиме 65
2.2. Загрузка системы на персональном компьютере 65
2.3. Утилиты начальной загрузки: LILO и GRUB 66
GRUB: универсальный загрузчик 67
LILO: традиционный загрузчик Linux 68
Параметры ядра 69
Мультисистемная загрузка на персональном компьютере 70
Мультисистемное конфигурирование GRUB 70
Мультисистемное конфигурирование LILO 71
2.4. Загрузка в однопользовательском режиме 71
Однопользовательский режим при использовании GRUB 72
Однопользовательский режим при использовании LILO 72
2.5. Работа со сценариями запуска системы 72
Демон `init` и уровни выполнения 73
Сценарии запуска Red Hat и Fedora 76
Сценарии запуска SUSE 78
Сценарии запуска Debian и Ubuntu 80
2.6. Перезагрузка и останов системы 80
Выключение питания 81
Команда `shutdown`: корректный способ останова системы 81
Команда `halt`: более простой способ останова 82
Команда `reboot`: быстрый перезапуск 82
Команда `telinit`: изменение уровня выполнения демона `init` 82
Команда `poweroff`: запрос на выключение питания 82
2.7. Упражнения 83

Глава 3. Сила привилегий 85

3.1. Владение файлами и процессами 85
3.2. Суперпользователь 87
3.3. Пароль суперпользователя 88
3.4. Как стать суперпользователем 89
Команда `su`: замена идентификатора пользователя 89
Утилита `sudo`: ограниченный вариант команды `su` 90
3.5. Другие псевдопользователи 92
Унаследованный владелец системных команд: bin 93
Владелец непривилегированных системных программ: daemon 93
Общий пользователь NFS: nobody 93
3.6. Упражнения 93

Глава 4. Управление процессами 95

4.1. Атрибуты процесса 95
Идентификатор процесса (PID) 96
Идентификатор родительского процесса (PPID) 96
Идентификатор пользователя (UID) и эффективный идентификатор

пользователя (EUID) 97
Идентификатор группы (GID) и эффективный идентификатор группы (EGID) 97
Приоритет и фактор уступчивости 98
Управляющий терминал 98
4.2. Жизненный цикл процесса 98
4.3. Сигналы 99
4.4. Отправка сигналов: команды `kill` и `killall` 102
4.5. Состояния процесса 103
4.6. Изменение приоритета выполнения: команды `nice` и `renice` 104
4.7. Текущий контроль процессов: команда `ps` 105
4.8. Улучшенный текущий контроль процессов: команда `top` 107
4.9. Файловая система `/proc` 108
4.10. Отслеживание сигналов и системных вызовов: команда `strace` 109
4.11. Процессы, вышедшие из-под контроля 110
4.12. Рекомендуемая литература 112
4.13. Упражнения 112

Глава 5. Файловая система 113

5.1. Имена файлов и каталогов 115
5.2. Монтирование и размонтирование файловой системы 116
5.3. Организация файловой системы 118
5.4. Типы файлов 120
Обычные файлы 122
Каталоги 122
Файлы байт-ориентированных и блочно-ориентированных устройств 122
Локальные сокеты 123
Именованные каналы 124
Символические ссылки 124
5.5. Атрибуты файлов 125
Биты режима 125
Биты setuid и setgid 126
Sticky-бит 126
Просмотр атрибутов файла 126
Команда `chmod`: изменение прав доступа 128
Команда `chown`: смена владельца и группы 129
Команда `umask`: задание стандартных прав доступа 130
Дополнительные флаги 131
5.6. Списки контроля доступа 132
Обзор ACL 132
Записи, определенные по умолчанию 135
5.7. Упражнения 136

Глава 6. Подключение новых пользователей 137

6.1. Файл `/etc/passwd` 138
Регистрационное имя 138
Зашифрованный пароль 140
Идентификатор пользователя 141
Идентификатор группы по умолчанию 142
Поле GECOS 142

Домашний каталог 143
Регистрационная оболочка 143
6.2. Файл /etc/shadow 143
6.3. Файл /etc/group 146
6.4. Подключение пользователей 147
Редактирование файлов /etc/passwd и /etc/shadow 148
Редактирование файла /etc/group 148
Задание пароля 149
Создание домашнего каталога пользователя 149
Копирование стандартных конфигурационных файлов 149
Назначение каталога для электронной почты 151
Проверка новой учетной записи 151
Запись статуса учетной записи и контактной информации о пользователе 152
6.5. Удаление пользователей 152
6.6. Отключение учетной записи 153
6.7. Управление учетными записями 153
6.8. Упражнения 155

Глава 7. Добавление жесткого диска 157

7.1. Дисковые интерфейсы 158
Интерфейс PATA 159
Интерфейс SATA 160
Интерфейс SCSI 161
Что лучше: SCSI или IDE? 165
7.2. Конструкция современного жесткого диска 165
7.3. Файловые системы Linux 167
Файловые системы ext2fs и ext3fs 167
Файловая система ReiserFS 168
Файловые системы XFS и JFS 168
7.4. Процедура добавления жесткого диска 169
Подключение диска 169
Форматирование диска 169
Разбивка жесткого диска на разделы и присвоение им меток 170
Создание файловых систем в разделах диска 172
Монтирование файловых систем 173
Настройка автоматического монтирования файловой системы 174
Создание раздела подкачки 175
7.5. HDPARM: установка параметров интерфейса IDE 176
7.6. Команда fsck: проверка и восстановление файловых систем 178
7.7. Добавление жесткого диска: пошаговое руководство 181
7.8. Расширенное управление дисками: RAID и LVM 185
Программная реализация RAID в среде Linux 186
Управление логическими томами 187
Пример конфигурации, использующей LVM и RAID 187
Действия по отношению к отказавшему диску 192
Перераспределение дискового пространства 193
7.9. Монтирование дисков USB 194
7.10. Упражнения 195

Глава 8. Периодические процессы 197
8.1. Демон `cron`: системный планировщик 197
8.2. Формат `crontab`-файлов 198
8.3. Управление `crontab`-файлами 200
8.4. Стандартные применения демона `cron` 201
Чистка файловой системы 201
Распространение конфигурационных файлов по сети 202
Ротация журнальных файлов 203
8.5. Другие планировщики: демоны `anacron` и `fcron` 203
8.6. Упражнения 204

Глава 9. Резервное копирование 205
9.1. Принципы резервного копирования 206
Создавайте резервные копии на одном компьютере 206
Маркируйте носители 206
Правильно выбирайте периодичность резервного копирования 207
Будьте осмотрительны при выборе архивируемых файловых систем 207
Старайтесь умещать каждодневные архивы на одном носителе 207
Создавайте файловые системы, объем которых меньше емкости резервного носителя 208
Храните носители вне рабочего помещения 208
Защищайте резервные копии 209
Активность файловой системы во время создания архива должна быть низкой 209
Проверяйте состояние своих носителей 209
Определите жизненный цикл носителя 210
Компонуйте данные с учетом резервного копирования 211
Будьте готовы к худшему 211
9.2. Устройства и носители, используемые для резервного копирования 211
Оптические носители: CD-R/RW, DVD±R/RW и DVD-RAM 212
Съемные жесткие диски (USB и FireWire) 213
Малые лентопротяжные устройства: 8-миллиметровые и DDS/DAT 214
Устройства DLT/S-DLT 214
Устройства AIT и SAIT 215
Устройства VXA/VXA-X 215
Устройства LTO 215
Системы с автоматической загрузкой носителей 215
Жесткие диски 216
Типы носителей 216
Что покупать 217
9.3. Команда `dump`: настройка режима инкрементного архивирования 218
Архивирование файловых систем 218
Схемы создания архивов 220
9.4. Команда `restore`: восстановление файлов из резервных копий 222
Восстановление отдельных файлов 222
Восстановление файловых систем 224
9.5. Архивирование и восстановление при модификации операционной системы 225
9.6. Другие архиваторы 225
Команда `tar`: упаковка файлов 226
Команда `cpio`: древний архиватор 226

Команда dd: манипулирование битами 227
9.7. Запись нескольких архивов на одну ленту 227
9.8. Bacula 228
Модель, используемая программой Bacula 229
Настройка программы Bacula 230
Установка базы данных и демонов Bacula 231
Конфигурирование демонов Bacula 231
Конфигурирование демона управления: файл bacula-dir.conf 232
Конфигурирование демона хранения: файл bacula-sd.conf 236
Конфигурирование консоли: файл bconsole.conf 237
Установка и конфигурирование демона управления файлами клиента 237
Запуск демонов Bacula 238
Добавление носителей в пулы 239
Выполнение архивирования вручную 239
Выполнение задания восстановления 241
Мониторинг и отладка конфигураций программы Bacula 244
Альтернативы программе Bacula 245
9.9. Коммерческие системы резервного копирования 246
ADSM/TSM 246
Veritas 247
Прочие программы 247
9.10. Рекомендуемая литература 247
9.11. Упражнения 247

Глава 10. Система Syslog и журнальные файлы 251

10.1. Методы обработки журнальных файлов 251
Уничтожение журнальных файлов 252
Ротация журнальных файлов 252
Архивирование журнальных файлов 254
10.2. Журнальные файлы Linux 254
Специальные журнальные файлы 257
Журнальная регистрация на уровне ядра и на этапе начальной загрузки 257
10.3. Утилита logrotate: управление журнальными файлами 259
10.4. Syslog: система регистрации событий 260
Альтернативы системе Syslog 261
Архитектура системы Syslog 261
Конфигурирование демона syslogd 262
Разработка схемы журнальной регистрации для конкретной организации 265
Примеры конфигурационных файлов 266
Автономный компьютер 266
Сетевой клиент 267
Центральный регистрационный хост 267
Пример выходной информации системы Syslog 268
Программы, использующие систему Syslog 269
Отладка системы Syslog 270
Использование функций системы Syslog в сценариях 270
10.5. Поиск полезной информации в журнальных файлах 272
10.6. Упражнения 273

Глава 11. Управление программным обеспечением и конфигурацией 275
11.1. Базовая инсталляция Linux 276
Загрузка по сети на персональном компьютере 277
Конфигурирование протокола PXE в Linux 277
Дистанционная загрузка на специализированных компьютерах 278
Kickstart: автоматизированный инсталлятор Red Hat 279
AutoYaST: автоматизированный инсталлятор SUSE 282
Инсталлятор Debian и Ubuntu 283
Инсталляция из главной системы 284
11.2. Бездисковые клиенты 285
11.3. Управление пакетами 286
Доступные системы управления пакетами 287
`rpm`: управление пакетами RPM 288
`dpkg`: управление пакетами Debian 289
11.4. Высокоуровневые системы управления пакетами 290
Хранилища пакетов 291
Служба Red Hat Network 292
APT: усовершенствованное средство управления пакетами 293
Конфигурирование 294
Пример файла `/etc/apt/sources.list` 295
Использование прокси-серверов для кэширования пакетов 296
Создание внутреннего сервера APT 296
Автоматизация работы утилиты 297
`yum`: управление выпусками для RPM 298
11.5. Управление изменениями 299
Создание резервных файлов 299
Формальные системы управления изменениями 300
RCS: система контроля изменений 301
CVS: система параллельных версий 304
Subversion: отдаем должное CVS 305
11.6. Локализация и конфигурирование 308
Организация локализации 308
Тестирование 310
Локальная компиляция 310
Распространение локализаций 311
Устранение проблем планирования 312
11.7. Средства управления конфигурацией 313
`cfengine`: компьютерная иммунная система 313
LCFG: крупномасштабная система конфигурирования 314
Arusha Project (ARK) 314
Template Tree 2: помощник `cfengine` 314
DMTF/CIM: общая информационная модель 315
11.8. Организация совместного использования программ через NFS 316
Пространства имен пакетов 316
Управление зависимостями 317
Сценарии упаковщика 318
Средства реализации 318
11.9. Рекомендуемые программы 319
11.10. Рекомендуемая литература 321
11.11. Упражнения 321

ЧАСТЬ II. Работа в сетях 323

Глава 12. Сети TCP/IP 325
12.1. TCP/IP и Internet 326
Краткий исторический экскурс 326
Кто сегодня управляет сетью Internet 327
Сетевые стандарты и документация 328
12.2. Семейство протоколов TCP/IP 329
12.3. Пакеты и их инкапсуляция 331
Канальный уровень 332
Стандарты формирования фреймов Ethernet 332
Стандарты кабелей Ethernet 332
Беспроводные сети 333
Максимальная единица передачи 333
Адресация пакетов 334
Порты 335
Типы адресов 336
12.4. IP-адреса 336
Классы IP-адресов 336
Подсети и маски 337
Кризис IP-адресации 340
CIDR: протокол бесклассовой междоменной маршрутизации 341
Выделение адресов 343
Частные адреса и система NAT 344
Адресация в стандарте IPv6 346
12.5. Маршрутизация 348
Таблицы маршрутизации 348
Переадресующие ICMP-пакеты 349
12.6. ARP: протокол преобразования адресов 350
12.7. Подключение компьютера к сети 352
Присвоение сетевых имен и IP-адресов 353
Команда `ifconfig`: конфигурирование сетевых интерфейсов 354
Команда `mii-tool`: настройка автоматического согласования и других параметров физической среды 357
Команда `route`: конфигурирование статических маршрутов 358
Стандартные маршруты 359
Конфигурирование DNS 360
Стек сетевых протоколов Linux 361
12.8. Сетевое конфигурирование в различных дистрибутивах 362
Сетевое конфигурирование в Red Hat и Fedora 362
Сетевое конфигурирование в SUSE 364
Сетевое конфигурирование в Debian и Ubuntu 365
12.9. DHCP: протокол динамического конфигурирования хостов 365
Программное обеспечение DHCP 366
Схема работы DHCP 367
DHCP-сервер ISC 367
12.10. Динамическое переконфигурирование и настройка 369
12.11. Безопасность 371
Перенаправление IP-пакетов 371
Переадресующие ICMP-пакеты 371

Маршрутизация “от источника” 371
Широковещательные ICMP-пакеты и другие виды направленных широковещательных сообщений 372
Подмена IP-адресов 372
Встроенные брандмауэры Linux 373
Виртуальные частные сети 373
Переменные ядра, связанные с безопасностью 374
12.12. Реализация системы NAT в Linux 374
12.13. PPP: протокол двухточечного соединения 375
Производительность протокола PPP 375
Подключение к сети посредством протокола PPP 376
Как заставить систему работать по протоколу PPP 376
Управление PPP-соединениями 376
Присвоение адреса 377
Маршрутизация 377
Безопасность 377
Диалоговые сценарии 378
Конфигурирование протокола PPP в Linux 378
12.14. Особенности сетевого конфигурирования в Linux 385
12.15. Рекомендуемая литература 386
12.16. Упражнения 387

Глава 13. Маршрутизация 389

13.1. Подробнее о маршрутизации пакетов 390
13.2. Демоны и протоколы маршрутизации 392
Дистанционно-векторные протоколы 393
Топологические протоколы 394
Метрики стоимости 395
Внутренние и внешние протоколы 395
13.3. Основные протоколы маршрутизации 396
RIP: протокол маршрутной информации 396
RIP-2: протокол маршрутной информации, версия 2 397
OSPF: открытый протокол первоочередного обнаружения кратчайших маршрутов 397
IGRP и EIGRP: протоколы маршрутизации внутренних шлюзов 398
IS-IS: протокол маршрутизации между промежуточными системами 398
MOSPF, DVMRP и PIM: протоколы многоадресной маршрутизации 398
Протокол обнаружения маршрутизаторов 398
13.4. Демон `routed`: стандартный демон маршрутизации 399
13.5. Демон `gated`: открытие неведомых возможностей 399
13.6. Выбор стратегии маршрутизации 400
13.7. Маршрутизаторы Cisco 401
13.8. Рекомендуемая литература 404
13.9. Упражнения 405

Глава 14. Сетевые аппаратные средства 407

14.1. ЛВС, ГВС и ОВС 408
14.2. Ethernet: общепризнанная ЛВС 408
Как работает Ethernet 410

Топология Ethernet 410
Неэкранированная витая пара 411
Соединение и расширение сетей Ethernet 413
Концентраторы 413
Коммутаторы 414
Маршрутизаторы 416
14.3. Беспроводные сети: ЛВС путешественников 416
Безопасность беспроводных сетей 417
Беспроводные коммутаторы 418
14.4. FDDI: дорогая и разочаровывающая ЛВС 418
14.5. ATM: локальная сеть несбывшихся надежд 420
14.6. Frame Relay: глобальная убыточная сеть 421
14.7. ISDN: глобальная сеть-невидимка 422
14.8. DSL и кабельные модемы: всенародная глобальная сеть 422
14.9. Будущее сетей 423
14.10. Тестирование и отладка сетей 424
14.11. Прокладка кабелей 424
Кабели НВП 424
Офисные точки подключения 425
Стандарты кабельных систем 425
14.12. Проектирование сетей 426
Структура сети и архитектура здания 427
Существующие сети 427
Расширение сетей 427
Перегрузка 428
Обслуживание и документирование 428
14.13. Управление сетью 428
14.14. Рекомендуемые поставщики 429
14.15. Рекомендуемая литература 430
14.16. Упражнения 430

Глава 15. Система доменных имен 431

15.1. DNS для нетерпеливых: подключение нового компьютера 432
15.2. История DNS 434
Реализации BIND 434
Другие реализации DNS 434
15.3. Основные задачи DNS 436
15.4. Пространство имен DNS 437
Владельцы доменов 440
Выбор имени домена 441
Лавинообразный рост числа доменов 441
Регистрация домена второго уровня 441
Создание собственных поддоменов 442
15.5. Принципы работы DNS 442
Делегирование 442
Кэширование и эффективность 443
Расширенный протокол DNS 444
15.6. Что нового в DNS 445

15.7. База данных DNS 448
Записи о ресурсах 448
Запись SOA 451
Записи NS 454
Записи A 454
Записи PTR 455
Записи MX 456
Записи CNAME 458
Специальное применение записей CNAME 458
Записи LOC 460
Записи SRV 461
Записи TXT 462
Записи о ресурсах IPv6 463
Прямое преобразование в стандарте IPv6 463
Обратное преобразование в стандарте IPv6 463
Записи, определяющие безопасность 464
Директивы в файлах зон 464
Связующие записи: ссылки между зонами 466
15.8. Пакет BIND 468
Версии пакета BIND 468
Определение текущей версии 469
Компоненты пакета BIND 470
Демон `named`: сервер имен пакета BIND 471
Авторитетные и кэширующие серверы 472
Рекурсивные и нерекурсивные серверы 472
Функции распознавания имен 473
Выполнение запросов из командной строки 474
15.9. Проектирование среды DNS 474
Управление пространством имен 475
Авторитетные серверы 475
Кэширующие серверы 476
Защита 477
Подводя итоги 477
Задачи, выполняемые с помощью DNS/BIND 477
15.10. Работа с клиентом BIND 478
Конфигурирование распознавателя 478
Тестирование распознавателя 479
Влияние на остальную часть системы 479
15.11. Конфигурирование сервера BIND 480
Аппаратные требования 480
Конфигурационные файлы 480
Инструкция `include` 482
Инструкция `options` 482
Инструкция `acl` 489
Инструкция `key` 489
Инструкция `trusted-keys` 490
Инструкция `server` 490
Инструкция `masters` 491
Инструкция `logging` 492
Инструкция `zone` 492

Инструкция controls 496
Расщепление DNS и инструкция view в BIND 9 497
15.12. Примеры конфигурации пакета BIND 498
Зона localhost 499
Небольшая компания, предоставляющая консалтинговые услуги в области безопасности 500
Internet System Consortium, isc.org 504
15.13. Запуск демона named 506
15.14. Обновление файлов зон 506
Зонные пересылки 507
Динамическое обновление 508
15.15. Вопросы безопасности 510
Еще раз о списках управления доступом 511
Ограничение возможностей демона named 512
Безопасные межсерверные взаимодействия посредством технологий TSIG и TKEY 513
Технология DNSSEC 516
Отрицательные ответы 522
Microsoft и DNS 523
15.16. Тестирование и отладка 525
Журнальная регистрация 525
Пример конфигурации регистрации в журнале 530
Уровни отладки 530
Отладка посредством команды rndc 531
Сбор статистических данных в BIND 532
Отладка с помощью команды dig 533
Напрасное делегирование 534
doc: контроль за "чистотой" домена 536
Другие средства контроля за "чистотой" DNS 537
Проблемы, связанные с производительностью 537
15.17. Особенности пакета BIND в различных дистрибутивах 538
15.18. Рекомендуемая литература 541
Списки рассылки 541
Книги и другая документация 541
Ресурсы в Internet 541
Документы RFC 542
15.19. Упражнения 542

Глава 16. Сетевая файловая система 543

16.1. Общая информация об NFS 543
Версии протокола NFS 544
Выбор транспортного протокола 544
Блокировка файлов 545
Дисковые квоты 545
Секретные ключи и монтирование без учета состояния 545
Соглашения об именах совместно используемых файловых систем 546
Безопасность и NFS 546
Учетные записи root и nobody 547
16.2. Серверная часть NFS 548
Файл exports 549
Демон nfsd: файловый сервер 551

16.3. Клиентская часть NFS 552
Монтирование файловых систем NFS на этапе начальной загрузки 554
Ограничение экспортирования в незащищенные порты 555
16.4. Команда `nfsstat`: отображение статистики NFS 555
16.5. Специализированные файловые серверы NFS 556
16.6. Автоматическое монтирование 557
`automount`: монтирование файловых систем по требованию 557
Главный файл 558
Таблицы назначений 559
Исполняемые файлы таблиц 559
16.7. Рекомендуемая литература 560
16.8. Упражнения 561

Глава 17. Совместное использование системных файлов 563

17.1. Предмет совместного использования 564
17.2. Демон `nscd`: кэширование результатов поиска 565
17.3. Копирование файлов 566
Утилита `rdist`: принудительная рассылка файлов 567
Утилита `rsync`: более безопасная рассылка файлов 570
Рассылка файлов по запросу 572
17.4. NIS: сетевая информационная служба 573
Схема работы NIS 574
Преимущества и недостатки NIS 576
Задание приоритетов для источников административной информации 577
Сетевые группы 579
Создание домена NIS 580
Настройка параметров управления доступом в файле `/etc/ypserv.conf` 581
О тонкостях NIS в разных дистрибутивах 582
17.5. LDAP: упрощенный протокол доступа к каталогам 582
Структура данных LDAP 583
Особенности LDAP 584
Документация и спецификации LDAP 585
Замена NIS на LDAP 587
LDAP и безопасность 588
17.6. Рекомендуемая литература 588
17.7. Упражнения 588

Глава 18. Электронная почта 591

18.1. Системы электронной почты 594
Пользовательские агенты 595
Транспортные агенты 596
Агенты доставки 596
Хранилища сообщений 596
Агенты доступа 597
Агенты подачи почты 597
18.2. Структура почтового сообщения 598
Адресация почты 599
Интерпретация заголовков почтовых сообщений 599

18.3. Принципы организации электронной почты 603
Почтовые серверы 604
Почтовые каталоги пользователей 606
Протоколы IMAP и POP 607
18.4. Почтовые псевдонимы 608
Загрузка списков рассылки из файла 611
Направление почты в файл 611
Направление почты в программу 612
Примеры псевдонимов 612
Переадресация почты 614
Хешированная база данных псевдонимов 615
18.5. Списки рассылки и программы для работы с ними 616
Программные пакеты для поддержки списков рассылки 616
LDAP 620
18.6. “Почтмейстер” sendmail 622
История программы sendmail 623
Инсталляция программы sendmail 624
Инсталляция программы sendmail в Debian и Ubuntu 626
Файл переключения служб 627
Режимы работы 627
Очередь почтовых сообщений 629
18.7. Конфигурирование программы sendmail 631
Использование препроцессора m4 632
Файлы для конфигурирования программы sendmail 633
Создание файла конфигурации из готового mc-файла 634
Изменение конфигурации программы sendmail 635
18.8. Базовые примитивы конфигурации программы sendmail 635
Макрос VERSIONID 636
Макрос OSTYPE 636
Макрос DOMAIN 638
Макрос MAILER 638
18.9. Дополнительные примитивы конфигурации программы sendmail 640
Макрос FEATURE 640
Средство use_cw_file 640
Средство redirect 641
Средство always_add_domain 641
Средство nocanonify 642
Таблицы и базы данных 642
Средство mailertable 644
Средство genericstable 644
Средство virtusertable 645
Средство ldap_routing 646
Маскирование адресов и макрос MASQUERADE_AS 647
Макроконстанты MAIL_HUB и SMART_HOST 649
Маскирование и маршрутизация 649
Средство nullclient 650
Средства local_lmtp и smrsh 651
Средство local_procmail 651
Макросы LOCAL_* 652
Конфигурационные опции 652

18.10. Средства программы sendmail для борьбы со спамом 654
Ретрансляция 656
База доступа 658
Черные списки 661
Проверка заголовков 661
Ограничение скорости и количества соединений 663
“Хлопанье дверью” 663
Фильтрация почты 664
Обработка спама 664
SpamAssassin 665
SPF и Sender ID 666
18.11. Примеры файлов конфигурации 666
Клиентские компьютеры на хосте sendmail.com 666
Главный сервер хоста sendmail.com 667
18.12. Безопасность и программа sendmail 670
Владельцы файлов 671
Права доступа 672
Безопасная пересылка почты в файлы и программы 673
Опции безопасности 674
Изменение корневого каталога программы sendmail 675
Отражение атак типа “отказ в обслуживании” 675
Фальсификации 676
Безопасность сообщений 677
SASL: простой протокол аутентификации и защиты 677
18.13. Производительность программы sendmail 678
Режимы доставки 678
Группы очередей и разбивка конвертов 679
Обработчики очередей 680
Контроль средней загруженности 681
Обработка недоставленных сообщений 681
Настройка ядра 682
18.14. Сбор статистических данных, тестирование и отладка 683
Тестирование и отладка 683
Доставка в протокольном режиме 685
Обмен данными по протоколу SMTP 685
Мониторинг очередей 686
Журнальная регистрация 687
18.15. Почтовая система Exim 689
История 689
Программа Exim в Linux 689
Конфигурирование программы Exim 690
Что общего между программами Exim и sendmail 690
18.16. Postfix 691
Архитектура Postfix 691
Получение почты 692
Администратор очередей 692
Отправка почты 693
Защита 693
Команды и документация Postfix 693
Конфигурирование Postfix 694

Что должно находиться в файле `main.cf` 694
Базовые настройки 695
Использование `postconf` 695
Таблицы преобразований 696
Локальная доставка 697
Виртуальные домены 698
Виртуальные домены псевдонимов 699
Виртуальные домены почтовых ящиков 700
Управление доступом 700
Таблицы доступа 701
Аутентификация клиентов 703
Борьба со спамом и вирусами 703
Черные списки 704
SpamAssassin и `procmail` 705
Демоны политики 705
Фильтрация содержимого 705
Отладка 706
Просмотр очереди 707
Мягкий рикошет 707
Проверка управления доступом 707
18.17. Рекомендуемая литература 708
18.18. Упражнения 710

Глава 19. Управление сетями 713

19.1. Поиск неисправностей в сетях 714
19.2. Команда `ping`: проверка доступности компьютера 715
19.3. Команда `traceroute`: трассировка IP-пакетов 717
19.4. Команда `netstat`: получение информации о состоянии сети 719
Анализ конфигурационных данных интерфейсов 720
Мониторинг состояния сетевых соединений 721
Выявление прослушивающих сетевых служб 722
Изучение таблицы маршрутизации 722
Просмотр статистических данных функционирования различных сетевых протоколов 723
19.5. SAR: анализ текущего состояния интерфейсов 724
19.6. Анализаторы пакетов 725
Утилита `tcpdump`: самый лучший анализатор 726
Wireshark: визуальный анализатор пакетов 727
19.7. Протоколы управления сетями 728
19.8. SNMP: простой протокол управления сетями 729
Структура протокола SNMP 729
Операции протокола SNMP 731
RMON: база MIB для дистанционного мониторинга 731
19.9. Агент NET-SNMP 732
19.10. Программы управления сетями 733
Команды пакета NET-SNMP 733
Сбор и графическое представление данных SNMP 734
Nagios: основанный на событиях мониторинг SNMP и служб 736
Коммерческие системы сетевого управления 737
19.11. Рекомендуемая литература 738
19.12. Упражнения 738

Глава 20. Безопасность 741
20.1. Безопасна ли ОС Linux? 742
20.2. Слабые места в системе защиты 743
Социальная инженерия 743
Уязвимые места в ПО 744
Конфигурационные ошибки 745
20.3. Сертификаты и стандарты 746
Сертификаты 747
Стандарты 748
20.4. Безопасность: ключевые аспекты и философия 749
Фильтрация пакетов 749
Ненужные службы 750
Программные “заплаты” 750
Резервные копии 750
Пароли 750
Бдительность 750
Общие принципы защиты 750
20.5. Проблемы защиты файлов `/etc/passwd` и `/etc/shadow` 751
Проверка и выбор паролей 752
Устаревание паролей 753
Групповые и совместно используемые учетные записи 753
Пользовательские интерпретаторы команд 754
Привилегированные учетные записи 754
Модули PAM 754
20.6. Возможности POSIX 756
20.7. Программы с установленным битом смены идентификатора пользователя 757
20.8. Специальные права доступа 757
20.9. Различные аспекты безопасности 758
Дистанционная регистрация сообщений 758
Защищенные терминалы 759
Файлы `/etc/hosts.equiv` и `~/.rhosts` 759
Безопасность и NIS 759
Безопасность и NFS 760
Безопасность и `sendmail` 760
Безопасность и резервное копирование 760
Вирусы и черви 760
Троянские кони 761
Руткиты 762
20.10. Инструментальные средства защиты 762
Nmap: сканирование сетевых портов 762
Nessus: сетевой сканер следующего поколения 764
Утилита John the Ripper: поиск ненадежных паролей 765
`hosts_access`: управление доступом хостов 765
Программа Samhain: выявление попыток проникновения на компьютерах 766
Пакет SELinux (Security-Enhanced Linux) 767
20.11. Системы криптографической защиты 769
Kerberos: унифицированный подход к сетевой безопасности 769
PGP: высокая конфиденциальность 770
SSH: безопасный интерпретатор команд 771

Одноразовые пароли 773
Программа Stunnel 773
20.12. Брандмауэры 776
Фильтрация пакетов 776
Принципы фильтрации служб 776
Фильтрация на уровне служб 778
Фильтрация с учетом состояния 778
Насколько безопасны брандмауэры? 779
20.13. Поддержка брандмауэров в Linux: IP-таблицы 779
20.14. Виртуальные частные сети 783
Туннели IPsec 783
Так ли уж нужны виртуальные частные сети? 784
20.15. Специальные “защищенные” дистрибутивы Linux 784
20.16. Что нужно делать в случае атаки на сервер 785
20.17. Источники информации, касающейся безопасности 786
Организация CERT 787
Сервер SecurityFocus.com и список рассылки BugTraq 787
Информационный бюллетень Crypto-Gram 787
Организация SANS 788
Информационные ресурсы отдельных дистрибутивов 788
Другие списки рассылки и Web-сайты 789
20.18. Рекомендуемая литература 789
20.19. Упражнения 790

Глава 21. Web-хостинг и серверы Internet 793

21.1. Основы Web-хостинга 794
Унифицированные указатели ресурсов 794
Принцип работы HTTP 795
Генерация содержимого на лету 796
Балансировка нагрузки 797
21.2. Инсталляция HTTP-сервера 798
Выбор сервера 798
Установка сервера Apache 799
Конфигурирование сервера Apache 800
Запуск сервера Apache 801
Анализ журнальных файлов 801
Оптимизация для обеспечения высокопроизводительного хостинга статического содержимого 802
21.3. Виртуальные интерфейсы 802
Использование виртуальных хостов, основанных на именах 803
Конфигурирование виртуальных интерфейсов 803
Уведомление Apache о наличии виртуальных интерфейсов 804
21.4. Протокол защищенных сокетов (SSL) 805
Генерация запроса на подпись сертификата 806
Конфигурирование сервера Apache так, чтобы он использовал SSL 807
21.5. Кэширование и прокси-серверы 808
Кэш Squid и прокси-сервер 808
Инсталляция сервера Squid 809
21.6. Установка анонимного FTP-сервера 810
21.7. Упражнения 812

ЧАСТЬ III. Разное 813

Глава 22. Система X Window System 815

22.1. Диспетчер дисплеев 817
22.2. Запуск X-приложения 818
Переменная окружения DISPLAY 819
Аутентификация клиентов 820
Перенаправление соединений с помощью протокола SSH 821
22.3. Конфигурирование X-сервера 823
Секция Device 825
Секция Monitor 825
Секция Screen 826
Секция InputDevice 827
Секция ServerLayout 828
22.4. Устранение неполадок и отладка 829
Специальные комбинации клавиш в системе X 829
Когда хороший X-сервер приходит в негодность 830
22.5. Краткое замечание о настольных средах 831
KDE 832
GNOME 833
Что лучше: GNOME или KDE? 833
22.6. Рекомендуемая литература 834
22.7. Упражнения 834

Глава 23. Печать 835

23.1. Принтеры становятся более сложными 836
23.2. Языки принтеров 837
PostScript 837
PCL 838
PDF 838
XHTML 839
PJL 839
Драйверы принтеров и поддержка ими PDL-языков 840
23.3. Архитектура CUPS 841
Печать документов 841
Просмотр и манипулирование очередями на печать 842
Множество принтеров 842
Экземпляры принтеров 842
Сетевая печать 843
Лежащий в основе CUPS протокол: HTTP 844
PPD-файлы 845
Фильтры 846
23.4. Администрирование сервера CUPS 847
Настройка сервера печати 848
Автоматическое конфигурирование принтеров 848
Конфигурирование сетевых принтеров 849
Примеры конфигурирования принтеров 849
Создание класса принтеров 850
Отключение принтеров для прохождения технического осмотра 850

Другие задачи, связанные с конфигурированием 851
Форматы бумаги 851
Команды совместимости 853
Основные программы печати 854
Документация по системе CUPS 855
23.5. Советы по выявлению проблем 855
Ведение журналов в CUPS 856
Проблемы с прямой печатью 856
Проблемы с сетевой печатью 856
Проблемы с дистрибутивами 857
23.6. Практические советы касательно принтеров 857
Выбор принтера 857
GDI-принтеры 858
Двусторонняя печать 858
Другие аксессуары для принтера 859
Принтеры с последовательным и параллельным интерфейсом 859
Сетевые принтеры 859
23.7. Другие советы касательно принтеров 860
Используйте титульные страницы только в случае необходимости 860
Обеспечивайте утилизацию отходов 860
Устанавливайте программы для предварительного просмотра 860
Покупайте дешевые принтеры 861
Всегда держите под рукой дополнительные картриджи 861
Обращайте внимание на то, во сколько обходится печать каждой страницы 862
Организуйте учет использования принтера 863
Защищайте принтеры 863
23.8. Печать в KDE 864
`kprinter`: печать документов 865
Konqueror и печать 866
23.9. Рекомендуемая литература 866
23.10. Упражнения 867

Глава 24. Обслуживание аппаратных средств и условия эксплуатации 869

24.1. Основы обслуживания аппаратных средств 870
24.2. Контракты на обслуживание 870
Обслуживание на месте 870
Обслуживание с заменой плат 871
Гарантийные обязательства 871
24.3. Уход за электронным оборудованием 871
Статическое электричество 872
Переустановка плат 872
24.4. Мониторы 872
24.5. Модули памяти 873
24.6. Профилактическое обслуживание 873
24.7. Условия эксплуатации 874
Температура 874
Влажность 875
Охлаждение офисных помещений 875
Охлаждение машинных залов 875
Контроль температуры 877

24.8. Источники питания 877
24.9. Стойки для аппаратуры 878
24.10. Стандарты для центров по обработке данных 878
24.11. Инструменты 879
24.12. Рекомендуемая литература 880
24.13. Упражнения 880

Глава 25. Анализ производительности 881

25.1. Способы повышения производительности 882
25.2. Факторы, влияющие на производительность 884
25.3. Проверка производительности системы 885
Анализ использования центрального процессора 885
Управление памятью в Linux 887
Анализ использования памяти 890
Анализ операций обмена с диском 892
Выбор планировщика ввода-вывода 894
`sar`: сбор статических данных и генерирование отчетов по ним 894
`oprofile`: универсальный профилировщик 895
25.4. Помогите! Моя система почти остановилась! 895
25.5. Рекомендуемая литература 897
25.6. Упражнения 897

Глава 26. Взаимодействие с Windows 899

26.1. Вход в систему Linux из Windows 899
26.2. Получение доступа к удаленным настольным средам 900
Запуск сервера X Server на компьютере Windows 901
VNC: технология виртуальных сетей 902
Windows RDP: протокол удаленных рабочих столов 903
26.3. Запуск Windows и Windows-приложений 904
Двойная загрузка и почему ею не стоит пользоваться 904
Альтернатива: OpenOffice.org 905
26.4. Использование утилит командной строки в Windows 905
26.5. Совместимость Windows со стандартами электронной почты и Web 906
26.6. Совместное использование файлов при помощи Samba и CIFS 906
Samba: сервер CIFS для UNIX 907
Инсталляция пакета Samba 908
Кодирование имен файлов 909
Обзор сетевого окружения 910
Аутентификация пользователей 911
Совместное использование основных файлов 912
Совместно используемые ресурсы 912
Прозрачная переадресация при помощи MS DFS 913
`smbclient`: простой клиент CIFS 914
Файловая система `smbfs` 914
26.7. Совместное использование принтеров при помощи Samba 916
Установка драйвера принтера из Windows 917
Установка драйвера принтера из командной строки 918
26.8. Отладка сервера Samba 919
26.9. Рекомендуемая литература 921
26.10. Упражнения 921

Глава 27. Устройства последовательного доступа 923
27.1. Разъем RS-232C 924
27.2. Альтернативные разъемные соединения 927
Мини-разъем DIN-8 927
Разъем DB-9 928
Разъем RJ-45 929
Стандарт Йоста для разъемных соединений RJ-45 930
27.3. Аппаратная и программная несущие 932
27.4. Аппаратный контроль передачи данных 932
27.5. Длина кабеля 933
27.6. Файлы последовательных устройств 933
27.7. Команда `setserial`: установка параметров последовательного порта 934
27.8. Конфигурирование программного обеспечения для устройств последовательного доступа 935
27.9. Конфигурирование аппаратных терминалов 935
Процедура регистрации в системе 935
Файл `/etc/inittab` 937
Поддержка терминалов: базы данных `termcap` и `terminfo` 938
27.10. Специальные символы и драйвер терминала 939
27.11. Команда `stty`: задание параметров терминала 940
27.12. Команда `tset`: автоматическое задание параметров терминала 941
27.13. Устранение зависания терминала 942
27.14. Модемы 942
Протоколы модуляции, коррекции ошибок и сжатия данных 943
Программа `minicom`: выходные звонки 944
Дуплексные модемы 944
27.15. Отладка последовательной линии 945
27.16. Другие распространенные порты ввода-вывода 945
Интерфейс USB 946
27.17. Упражнения 947

Глава 28. Драйверы и ядро 949
28.1. Адаптация ядра 950
28.2. Драйверы и файлы устройств 951
Файлы и номера устройств 951
Создание файлов устройств 952
`sysfs`: доступ к “сердцу” устройств 953
Соглашения об именовании устройств 953
28.3. Зачем и как конфигурировать ядро 954
28.4. Конфигурирование параметров ядра Linux 955
28.5. Сборка ядра Linux 957
Не чините то, что еще не поломано 958
Конфигурирование параметров ядра 958
Компиляция ядра 959
28.6. Добавление драйвера устройства в Linux 960
Распознание устройства ядром 962
28.7. Загружаемые модули ядра 962
28.8. “Горячее” подключение 964

28.9. Настройка параметров начальной загрузки 964
28.10. Рекомендуемая литература 965
28.11. Упражнения 965

Глава 29. Демоны 967

29.1. Демон `init`: первичный процесс 968
29.2. Демоны `cron` и `atd`: планирование запуска команд 969
29.3. Демоны `xinetd` и `inetd`: управление демонами 969
Конфигурирование демона `xinetd` 970
Конфигурирование демона `inetd` 972
Файл `services` 974
Демон `portmap`: отображение номеров RPC-служб на номера портов TCP и UDP 975
29.4. Демоны ядра 975
`klogd`: чтение сообщений ядра 976
29.5. Демоны печати 976
Демон `cupsd`: планировщик для системы CUPS 976
Демон `lpd`: управление процессом печати 977
29.6. Демоны обслуживания файлов 977
`rpc.nfsd`: обслуживание файлов 977
`rpc.mountd`: обработка запросов на монтирование 977
`amd` и `automount`: монтирование файловых систем по запросу 977
`rpc.lockd` и `rpc.statd`: управление блокировками NFS 978
`rpciod`: кэширование блоков данных NFS 978
`rpc.rquotad`: дистанционное управление квотами 978
`smbd`: файловый сервер и сервер печати для клиентов Windows 978
`nmbd`: сервер имен NetBIOS 978
29.7. Демоны административных СУБД 978
`ypbind`: поиск серверов NIS 979
`ypserv`: сервер NIS 979
`rpc.ypxfrd`: пересылка баз данных NIS 979
`lwresd`: упрощенный сервер библиотечного распознавателя 979
`nscd`: демон кэширования для службы имен 979
29.8. Демоны электронной почты 979
`sendmail`: транспортировка электронной почты 979
`smtpd`: демон протокола SMTP 980
`popd`: базовый сервер почтовых ящиков 980
`imapd`: высококлассный сервер почтовых ящиков 980
29.9. Демоны удаленной регистрации и удаленного выполнения команд 980
`sshd`: безопасный сервер удаленной регистрации 980
`in.rlogind`: устаревший сервер удаленной регистрации 981
`in.telnetd`: еще один сервер удаленной регистрации 981
`in.rshd`: сервер удаленного выполнения команд 981
29.10. Демоны начальной загрузки и конфигурирования 981
`dhcpd`: динамическое назначение адресов 982
`in.tftpd`: простейший сервер пересылки файлов 982
`rpc.bootparamd`: усовершенствованная поддержка бездисковых систем 982
`hald`: демон уровня аппаратных абстракций 982
`udevd`: сериализация уведомлений о подключении устройств 982

29.11. Другие сетевые демоны 983
talkd: сервер программы talk 983
snmpd: сервер дистанционного управления сетями 983
ftpd: сервер пересылки файлов 983
rsyncd: синхронизация файлов между группой компьютеров 983
routed: ведение таблиц маршрутизации 984
gated: ведение сложных таблиц маршрутизации 984
named: сервер DNS 984
syslogd: обработка журнальных сообщений 984
in.fingerd: поиск пользователей 984
httpd: Web-сервер 985
29.12. NTPD: демон синхронизации времени 985
29.13. Упражнения 986

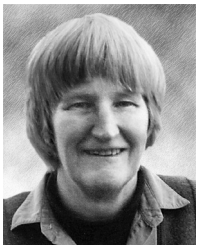
Глава 30. Стратегии и методики управления и администрирования 987

30.1. Что необходимо для того, чтобы все были довольны 988
30.2. Компоненты функциональной IT-организации 989
30.3. Роль управления 990
Руководство 991
Набор, увольнение и управление персоналом 991
Назначение заданий и контроль за их выполнением 995
Руководство вышестоящим начальством 996
Разрешение конфликтов 997
30.4. Роль администрирования 999
Продажа 999
Закупка 1000
Бухучет 1001
Кадры 1001
Маркетинг 1002
Дополнительные обязанности администратора 1003
30.5. Роль разработки 1003
Архитектурные принципы 1004
Анатомия системы управления 1006
Набор инструментов системного администратора 1007
Принципы разработки программного обеспечения 1008
30.6. Роль операций 1009
Обеспечение минимального времени простоя 1009
Документирование зависимостей 1010
Переналадка или списывание старого оборудования 1010
30.7. Работа отдела поддержки 1011
Доступность 1011
Объем услуг 1012
Перечень навыков 1014
Управление временем 1015
30.8. Документация 1015
Стандартизированная документация 1016
Маркировка оборудования 1017
Документация для пользователей 1019

30.9. Системы отслеживания запросов и уведомления о проблемах 1019
Общие функции диагностических систем 1020
Отношение пользователей к диагностическим системам 1020
Диагностические системы 1021
Распределение сообщений о неисправностях 1022
30.10. Восстановление после аварий 1023
Резервное копирование и хранение информации вне сети 1024
Подбор персонала на случай аварии 1025
Электропитание и кондиционирование 1025
Сетевая избыточность 1027
Проблемы с безопасностью 1027
Истории из “вторых рук”, связанные со Всемирным торговым центром 1028
30.11. Письменные стратегии 1029
Правила безопасности 1031
Соглашения о правилах работы пользователей 1032
Соглашения о правилах работы системных администраторов 1034
30.12. Правовые вопросы 1035
Шифрование 1036
Авторские права 1036
Конфиденциальность 1038
“Прощелкивание” EULA-соглашений 1040
Внедрение стратегии в жизнь 1041
Контроль = ответственность 1041
Лицензии на программное обеспечение 1043
Регуляторная политика 1043
30.13. Патенты на программное обеспечение 1044
30.14. Стандарты 1046
LSB: база стандартов Linux 1047
POSIX 1047
ITIL: библиотека интерфейсов информационных технологий 1047
COBIT: задачи по управлению информационными и другими связанными с ними технологиями 1048
30.15. Идеология Linux 1049
30.16. Основные поставщики Linux 1050
30.17. Организации, конференции и другие ресурсы 1053
Конференции и торговые выставки 1054
LPI: институт профессионалов Linux 1056
Списки рассылки и Web-ресурсы 1056
Опросы системных администраторов 1057
30.18. Рекомендуемая литература 1058
Инфраструктура 1058
Управление 1058
Политика и безопасность 1059
Правовые вопросы, патенты и конфиденциальность 1059
Общие индустриальные новости 1059
30.19. Упражнения 1059

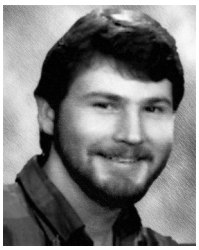
Предметный указатель 1061

Об авторах



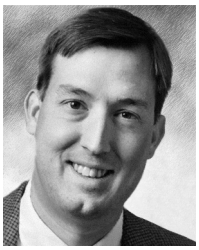
Эви Немет уволилась с факультета вычислительной техники университета штата Колорадо, но по-прежнему остается внештатным сотрудником CAIDA (Cooperative Association for Internet Data Analysis — совместная ассоциация по анализу данных в сети Internet) в центре суперкомпьютеров в Сан-Диего. В настоящее время она бороздит карибские просторы на своей новой игрушке — 40-футовой яхте "Wonderland" ("Страна чудес").

sailingevi@yahoo.com



Гарт Снайдер работал в компаниях NeXT и Sun и получил степень бакалавра электротехники в колледже Суортмор, штат Пенсильвания, а также ученую степень в университете Рочестера, штат Нью-Йорк.

garth@grsweb.us



Трент Р. Хейн — один из основателей компании Applied Trust Engineering, которая разрабатывает средства защиты и анализа производительности сетей. Трент получил степень бакалавра вычислительной техники в университете штата Колорадо.

trent@atrust.com

О помощниках

Линда Мак-Гинли (Linda McGinley) на протяжении 20 лет занимается системным и сетевым администрированием в учебной среде, и привлекалась к написанию третьего издания книги *UNIX System Administration Handbook*. В настоящее время она работает инженером по безопасности в Университетской корпорации атмосферных исследований (University Corporation for Atmospheric Research).

Бен Вэйли (Ben Whaley) — главный инженер в Applied Trust Engineering. Он получил диплом по специальности "Вычислительная техника" в Университете Колорадо, а еще является сертифицированным инженером Red Hat. В дополнение к программированию на Perl и PHP, Бен — наставник в программе Big Brother Big Sister (Старший брат, старшая сестра).

Адам Боггз (Adam Boggs) до конца 90-х гг. был студентом-администратором в одной из студенческих лабораторий Университета Колорадо. Несколько последних лет он принимал участие в разработке ядра, занимаясь написанием ПО файловых систем и хранилищ данных для Solaris и Linux. В настоящее время Адам работает администратором се-

тей, хранилищ данных и кластеров на суперкомпьютерах, используемых в атмосферном моделировании.

Джефри С. Химер (Jeffrey S. Haemer) (jsh@usenix.org) занимался коммерческими работами, связанными с UNIX и Linux, начиная с 1983 г., и до сих пор не пользуется Microsoft Word. В течение 11 лет он работал на различных должностях у крупного производителя принтеров QMS. Эви обучала его вождению. Его любимый палиндром: "A man, a plan, a canal, Suez".

Тоби Отикер (Tobi Oetiker) (tobi@oetiker.ch) — инженер-электрик по образованию и системный администратор по призванию. В области системного администрирования он работает с 1994 г. После обустройства роскошной компьютерной среды для студентов, работы в течение нескольких лет в ETH Zurich и написания ряда популярных приложений с открытым кодом вроде MRTG, RRDtool и SmokePing, в настоящее время Тоби трудится в компании Oetiker+Partner AG, которая занимается предоставлением информационных и консультационных услуг. Женат и проживает в Швейцарии. Дополнительную информацию о Тоби можно найти на сайте tobi.oetiker.ch.

Фриц Цокер (Fritz Zaucker) — физик по образованию. После нескольких лет исследований в области изменения климата, последующие 10 лет он возглавлял группу IT-поддержки (IT Support Group, ISG.EE) в департаменте информационных технологий и электротехники в ETH Zurich. Вместе с группой сотрудников он занимался преобразованием ISG.EE в профессиональную организацию, ориентированную на эффективное системное администрирование, надежную IT-инфраструктуру и удовлетворение конечных потребителей. Сейчас он присоединился к своим коллегам в компании Oetiker+Partner AG, которая занимается предоставлением информационных и консультационных услуг. Фрица можно найти на www.zauker.ch. Он ходил на яхте вместе с Эви и своими племянницами по Средиземному морю и искренне рекомендует поучаствовать в этом.

Скотт Сидел (Scott Seidel) — главный инженер в Applied Trust Engineering, где специализируется на безопасности уровня предприятия и управлении производительностью. Он получил диплом по деловому администрированию в области информационных систем и финансов в Университете Колорадо. Исследовательские интересы Скотта включают системный мониторинг и виртуализацию серверов. В свободное от системного администрирования время его можно найти на кухне готовящим что-нибудь вкусненькое.

Брайан Буус (Bryan Buus) — вице-президент по разработке в Adeptive Software, компании, занимающейся созданием заказного программного обеспечения. До провала движения Internet-компаний он занимал пост вице-президента по разработке в XOR, подразделении Web- и Internet-служб. Брайан получил дипломы бакалавра и магистра по специальности "Вычислительная техника" в Бостонском университете.

Нед Мак-Клейн (Ned McClain) (ned@atrust.com) — один из основателей и руководитель технического отдела в Applied Trust Engineering. Он читает лекции по различным темам системного администрирования и безопасности на таких конференциях, как USENIX в США и APRICOT в Азии. Нед получил диплом по специальности "Вычислительная техника" в техническом колледже при Корнеллском университете и имеет сертификат CISSP #39389.

Девид Швикерт (David Schweikert) работает системным администратором в ETH Zurich (Швейцария), где, помимо прочего, отвечает за почтовую систему департамента информационных технологий и электротехники. Он является разработчиком проектов с открытым кодом Mailgraph (инструментальное средство для визуализации статистики электронной почты) и Postgrey (реализация Postfix). Девид также интересуется базами данных и системным администрированием крупных сайтов.

Предисловие к первому изданию

Я был весьма заинтригован, когда мне принесли новое издание книги *UNIX System Administration Handbook*, посвященное исключительно операционной системе Linux. В третьем издании рассматривалась система Red Hat Linux, но лишь как один из четырех вариантов UNIX. В новой версии книги описаны ключевые дистрибутивы Linux и изъят материал, не имеющий отношения к Linux. Было любопытно узнать, сколько изменений для этого пришлось внести.

Как оказалось, очень много. Дистрибутивы Linux создаются на основе единого семейства приложений с открытым исходным кодом, поэтому они гораздо ближе друг другу, чем все остальные разновидности UNIX. В результате многие места книги приобрели более конкретный характер. Вместо того чтобы рассказывать о *возможном* поведении системы, авторы теперь описывают ее *точное* поведение.

В то же время очевидно, что в книге представлено все богатство и многообразие программного обеспечения UNIX. Сегодня практически все популярные программы работают в среде Linux, и администраторы Linux-систем все реже сталкиваются с необходимостью прибегать к компромиссам. После того как лидеры компьютерной индустрии, в частности IBM, Oracle и Silicon Graphics, начали осваивать мир Linux, эта операционная система стала стремительно превращаться в универсальный стандарт, на который равняются остальные версии UNIX (и далеко не всегда успешно).

Как убедительно доказывают авторы книги, Linux-системы столь же функциональны, безопасны и надежны, как и их коммерческие аналоги. Благодаря постоянным усилиям тысяч разработчиков операционная система Linux выходит на передовые позиции в компьютерном мире. Авторы книги прекрасно знают эту область, поэтому я смело оставляю вас с ними наедине. Наслаждайтесь приобретением знаний!

Линус Торвальдс
апрель 2002 г.

Введение

Когда мы написали первое издание этой книги (около 5 лет тому назад), ОС Linux только начинала совершенствоваться для нужд корпоративного мира. Мы надеемся, что наша книга поможет ощутить тот факт, что Linux становится операционной системой первого уровня, возможности которой полностью соответствуют требованиям таких крупнейших корпораций, как Sun, HP и IBM.

Сейчас Linux занимается IBM. Все, кто ждал недвусмысленного сигнала о том, что "Linux-воды стали безопасными для корпоративных пловцов", обнародованный корпорацией IBM в 2004 г. анонс, касающийся поддержки Linux в своей линейке серверов, могут считать, что такой сигнал получен. С IBM еще никто не пропадал, а Linux в наши дни является, в общем и целом, надежным решением[1].

Мы решили написать книгу, которая была бы настольным пособием профессионального системного администратора Linux. Там, где это было возможно, мы адаптировали проверенные методики и материалы нашей популярной книги *UNIX System Administration Handbook*. Было добавлено много нового материала, касающегося исключительно Linux, много было изменено, но в целом структура книги осталась той же. Надеемся, читатели согласятся с нами, что результат представляет собой высококлассное руководство по администрированию Linux-систем, в котором собрано все самое ценное из предыдущих изданий.

Существуют и другие книги по этой тематике, но ни в одной из них глубина изложения материала недостаточна для эффективной эксплуатации Linux-систем в реальных коммерческих условиях. Вот чем выгодно отличается наше издание.

- Книга является практическим руководством, в котором мы делимся с читателями нашим коллективным опытом системного администрирования. Она содержит множество практических примеров и советов.
- Это книга не о том, как использовать Linux в домашних условиях. Мы описываем применение операционной системы в коммерческих компаниях, правительственных учреждениях, университетах.
- В книге подробно излагаются вопросы, связанные с работой Linux-систем в сетях. Это самый трудный аспект системного администрирования, и именно здесь наша помощь, как нам кажется, будет для читателей наиболее полезной.
- Материал книги не подается в упрощенном виде. Примеры в большинстве случаев взяты из практики эксплуатации реальных систем со всеми их подводными камнями и нюансами.
- В книге рассмотрены пять основных дистрибутивов Linux.

Демонстрационные дистрибутивы

Подобно многим операционным системам, Linux развивается в нескольких направлениях. Несмотря на то что разработка ядра осуществляется на удивление централизованно, комплектация и распространение различных вариантов операционной системы находятся под контролем целого ряда организаций, каждая из которых преследует собственные цели.

[1] Во всяком случае, для серверов. В наши дни полем битвы являются настольные системы — область, где позиции Microsoft Windows чрезвычайно сильны. Исход этой борьбы все еще предсказать трудно. На момент написания этой книги система Windows по-прежнему предлагает более изящный интерфейс пользователя. Однако вряд ли можно ее назвать "свободной".

В книге рассматриваются пять дистрибутивов Linux:

- Red Hat® Enterprise Linux® 4.3 ES,
- Fedora™ Core 5,
- SuSE® Linux Enterprise 10.2,
- Debian® GNU/Linux 3.2 "Etch" (тестовый выпуск от сентября 2006 г.),
- Ubuntu® 6.06 "Dapper Drake".

Мы выбрали эти дистрибутивы потому, что они относятся к числу наиболее популярных и позволяют продемонстрировать весь спектр подходов к администрированию Linux-систем. В то же время большая часть материала книги применима и к другим дистрибутивам общего назначения.

При рассмотрении каждой темы мы даем подробную информацию об этих дистрибутивах. Комментарии, относящиеся к тому или иному дистрибутиву, отмечены в книге соответствующими значками.

Структура книги

Книга разбита на три большие части:

- "Основы администрирования";
- "Работа в сетях";
- "Разное".

В первой части приводится общий обзор Linux, сделанный с точки зрения системного администратора. В главах этой части представлены основные сведения и описаны методики, необходимые для управления автономной Linux-системой.

В части "Работа в сетях" описаны протоколы, используемые в Linux-системах, а также способы построения, расширения и администрирования сетей. Здесь же рассматривается высокоуровневое сетевое программное обеспечение. Среди изучаемых тем можно выделить систему доменных имен (DNS), сетевую файловую систему (NFS), средства маршрутизации, систему электронной почты `sendmail` и инструменты сетевого управления.

Часть "Разное" содержит вспомогательную информацию. Здесь описываются дополнительные программные пакеты, в частности система печати в Linux, даются рекомендации по обслуживанию аппаратных средств, принципам администрирования Linux-систем и т.д.

В конце каждой главы приводится набор практических заданий. Каждому упражнению дана оценка уровня сложности, причем учитывается как сложность самой задачи, так и время, затрачиваемое на ее выполнение.

Упражнения соответствуют четырем уровням сложности.

Звездочки отсутствуют	Простое задание, не требующее особых усилий
☆	Более сложное или трудоемкое задание, может потребоваться лабораторная работа
☆☆	Наиболее сложное или трудоемкое задание, требующее лабораторной работы
☆☆☆☆☆	Курсовые проекты (лишь в нескольких главах)

Для выполнения некоторых упражнений требуется доступ к системе с правами суперпользователя или посредством `sudo`. Иногда нужно получить предварительное разрешение у системного администратора локальной сети. Подобные случаи оговариваются в задании.

Наши помощники

Мы признательны Адаму Боггзу (Adam Boggs), Брайану Буусу (Bryan Buus) и Неду Мак-Клейну (Ned McClain) за их вклад в создание книги. В этом издании также поучаствовали наши друзья Бен Вэйли (Ben Whaley), Тоби Отикер (Tobi Oetiker), Фриц Цокер (Fritz Zaucker), Джефри С. Химер (Jeffrey S. Haemer), Девид Швикерт (David Schweikert) и Скотт Сидел (Scott Seidel). Их глубокие знания в самых разных областях существенно обогатили материал книги. Кроме того, мы благодарим Линду Мак-Гинли (Linda McGinley), которая в дополнение к тому, что имеет отношение к значительному объему текста книги, еще и неутомимо трудилась над организацией нашей совместной работы.

Информация для контактов

Свои предложения и комментарии присылайте по адресу `linux@book.admin.com`. Мы отвечаем на большинство писем, но, пожалуйста, будьте терпеливы: иногда проходит несколько дней, прежде чем у кого-то из нас появляется возможность ответить на письмо. Чтобы получить список выявленных на данный момент опечаток, а также другую свежую информацию по книге, посетите наш Web-сайт `www.admin.com`.

Надеемся, что книга вам понравится, и желаем удачи в системном администрировании.

Эви Немет
Гарт Снайдер
Трент Р. Хейн

октябрь 2006 г.

Благодарности

Множество людей тем или иным образом помогли нам в работе над книгой.

Кто-то из них выполнял научное редактирование или предлагал интересные примеры, кто-то оказывал моральную поддержку. Эти люди заслуживают особой благодарности:

Бо Конелл (Bo Connell)	Джон Корбет (Jon Corbet)	Джим Лейн (Jim Lane)
Сэм Лефлер (Sam Leffler)	Крикет Лиу (Cricket Liu)	Дерек Мартин (Derek Martin)
Ласло Немет (Laszlo Nemeth)	Эрик Робинзон (Eric Robinson)	Сэм Столлер (Sam Stoller)
Пол Викси (Paul Vixie)	Аарон Вебер (Aaron Weber)	Грег Вудс (Greg Woods)

Наши редакторы, Катерин Нолан (Catherine Nolan) и Мэри Франц (Mary Franz), заслуживает не только благодарности, но и награды за то, что мужественно выдерживала капризы авторов. Они были бесконечно терпеливы к нам, даже когда мы этого не заслуживали, и сделали все возможное, чтобы заставить нас сосредоточиться на повышении качества материала.

Мэри Лу Нор (Mary Lou Nohr) проделала огромную работу по оформлению настоящего издания. Мы высоко ценим ее усилия.

Благодарим Марка Собелла (Mark G. Sobell) за трудную и терпеливую работу над предметным указателем. Мы были приятно поражены результатом.

Наконец, Эви хотела бы выразить благодарность и принести извинения многим пляжным барам и кафе на берегу Карибского моря, чьи свободные беспроводные соединения она похищала, когда причаливала яхту в местах максимальной мощности сигнала. Если бы она не трудилась над главами книги, не обращая внимания на этот рай вокруг, второе издание было бы последним. Но кто знает, вдруг она шутит?

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг.

Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

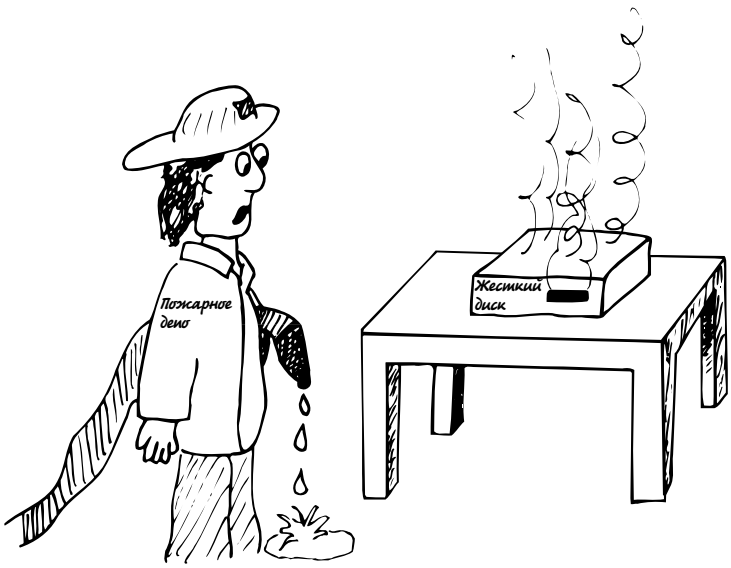
Информация для писем из:

России: 127055, Москва, ул. Лесная, д. 43, стр. 1

Украины: 03150, Киев, а/я 152

ЧАСТЬ I

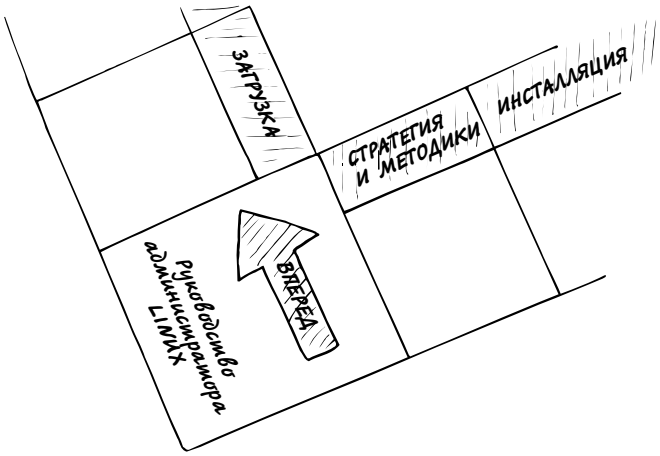
Основы администрирования



Book_LinuxAdmHand.indb 40 17.04.2007 13:54:30

Глава 1

С чего начать



Мы задались целью написать книгу, которая была бы надежным помощником системного администратора и служила источником практических советов и полезных сведений по теории системного администрирования, ведь их далеко не всегда можно найти на страницах интерактивного руководства. Таким образом, наша книга дополняет, но ни в коем случае не заменяет существующую систему документации к операционной системе Linux.

Эта книга поможет читателям:

- узнать о различных компонентах основных систем администрирования и понять принципы их совместной работы;
- ознакомиться с общими методами администрирования, которые необходимы в практической работе и проверены многолетним опытом авторов;
- научиться внедрять такие решения, которые в дальнейшем не помешают расширять и усложнять структуру системы;
- научиться отделять хорошие идеи от плохих и разбираться в особенностях различных системных дистрибутивов;
- усвоить комплекс основных приемов работы, что избавит читателя от необходимости рыться в документации, пытаясь узнать, как выполнить простое административное действие.

Перечисленные задачи невозможно решить с абсолютной степенью объективности, поэтому по ходу изложения мы постарались максимально пояснить свои субъективные взгляды и предпочтения. Особенность системного администрирования заключается в

том, что опытные администраторы могут иметь совершенно разные представления о правилах и процедурах управления системами. Читателям придется самостоятельно решать, какой именно материал и в какой степени соответствует той среде, в которой они работают.

1.1. Что необходимо знать

Мы предполагаем, что у читателей есть определенный опыт работы с Linux или UNIX. В частности, необходимо иметь общее представление о системе с точки зрения пользователя. Лишь при этом условии можно переходить к изучению принципов администрирования. Книги, перечисленные в разделе 1.10, помогут заложить необходимый фундамент знаний.

Большинство задач администрирования решается путем редактирования конфигурационных файлов и написания сценариев, поэтому читатели должны быть знакомы с каким-нибудь текстовым редактором. К разочарованию многих, использование программы Microsoft Word в качестве единственно возможного текстового редактора лишь усложнит и без того нелегкую жизнь администратора.

Настоятельно рекомендуем изучить редактор `vi` (модернизированная версия которого, `vim`, наиболее часто встречается в системах Linux). Он является стандартным для всех Linux-систем и, хотя выглядит несколько "бледным" на фоне более мощных программ (в частности, `emacs`), предоставляет достаточно большие и полные возможности. Нам также нравится редактор `pico`, который прост и прекрасно подходит новичкам, к тому же он включен во многие дистрибутивы.

Остерегайтесь нестандартных редакторов. Если отдать предпочтение одному из них, очень быстро надоест устанавливать его в каждой новой системе.

Одним из важнейших инструментов системного администратора являются сценарии для автоматизации основных задач. Примеры такого рода сценариев приводятся на протяжении всей книги. Чтобы стать профессионалом, необходимо научиться читать и модифицировать сценарии, написанные на языке интерпретаторов `Perl` и `sh` (в Linux его эквивалент — интерпретатор `bash`). Сценарии, которые пишутся "с нуля", можно составлять на командном языке или любом доступном языке сценариев.

📖 На Web-сайте `www.cpan.org` можно найти перечень полезных Perl-сценариев.

Для новых проектов мы рекомендуем применять язык Perl или Python. Как язык программирования он несколько необычен, однако содержит много средств, необходимых администраторам. В качестве учебника по этому языку советуем прочесть книгу Ларри Уолла *Programming Perl*; она также представляет собой образец профессионального технического руководства.

Полная ее библиографическая ссылка приведена в разделе 1.10.

Многие администраторы предпочитают иметь дело с языком Python. В определенном смысле это более элегантный язык, чем Perl, и его сценарии обычно легче читать и сопровождать. Статьи, посвященные сравнению языка Python с другими языками (включая Perl), можно найти по следующему адресу:

```
www.python.org/doc/Comparisons.html
```

Мы рекомендуем также изучить `expect`, который представляет собой скорее интерфейс для управления интерактивными программами, а не язык программирования. Освоить его можно достаточно быстро.

1.2. Взаимосвязь Linux и UNIX

Употребление названий Linux и UNIX в одном предложении чем-то сродни хождению по политическому минному полю или же попаданию в середину огромного участка зыбучих песков. Тем не менее, мы возьмем на себя смелость максимально лаконично и объективно изложить ряд фактов.

Linux — измененная и усовершенствованная реализация операционной системы UNIX. Она соответствует стандарту POSIX, работает на различных аппаратных платформах и совместима с большинством существующих приложений UNIX. Отличие от множества других вариантов UNIX заключается в том, что Linux — бесплатная операционная система с открытым исходным кодом, разрабатываемая коллективными усилиями тысяч энтузиастов и организаций. Вместе с тем, операционная система Linux содержит ряд технических усовершенствований, которые отсутствуют в первоначальных версиях UNIX, и поэтому представляет собой нечто большее, чем простой клон UNIX. Кроме того, она совершенно законно может считаться отдельным программным продуктом, который уже неправильно называть "UNIX". Следует заметить, что Linux — не единственная свободно доступная UNIX-подобная операционная система. Системы FreeBSD, NetBSD и OpenBSD, являющиеся потомками BSD UNIX, имеют своих пылких сторонников. Эти операционные системы сопоставимы с Linux по функциональным возможностям и надежности, хотя и не пользуются такой популярностью у сторонних разработчиков ПО.

Программы, написанные для Linux, являются UNIX-программами. Во многом благодаря проекту GNU наиболее значительные программные компоненты, составляющие основу UNIX-систем, разрабатываются теперь в соответствии с принципом открытости исходного кода. Один и тот же код может выполняться как в Linux, так и в любой другой однотипной среде.

Web-серверу Apache, к примеру, все равно, где работать: в Linux или в HP-UX. С позиции коммерческих приложений Linux представляет собой всего лишь одну из наиболее поддерживаемых версий UNIX.

Системы UNIX и Linux уже многие годы функционируют в промышленных условиях. В нашей книге, в отличие от многих других, посвященных этой теме, рассказывается о принципах эффективного применения Linux в крупномасштабных производственных средах, а не в качестве однопользовательских рабочих станций[1].

1.3. Linux в историческом контексте

Операционная система Linux появилась в 1991 г. как персональный проект Линуса Торвальдса, финского аспиранта. Первоначально он намеревался всего лишь модифицировать Minix — модельную операционную систему, написанную Эндрю Таненбаумом. Однако проект вызвал огромный интерес во всем мире, и вскоре ядро системы зажило своей жизнью. Благодаря коллективным усилиям сообщества разработчиков операционная система Linux стала стремительно эволюционировать. Ядро версии 1.0 было выпущено в 1994 г.; на момент написания книги (сентябрь 2006 г.) самая последняя стабильная версия ядра имела номер 2.6.17.

[1] Под "производственной" средой мы понимаем среду, которую организация использует для выполнения реальных задач (в отличие от использования для тестирования, исследований или разработки).

Поскольку Linux многим обязана своим предшественникам — разнообразным версиям UNIX, провозглашение наступления эры Linux в 1991 г. было несколько преждевременным. История UNIX началась несколько десятилетий назад, в 1969 г., в рамках научно-исследовательского проекта лабораторий Bell Labs корпорации AT&T. В 1976 г. система стала бесплатно распространяться в университетской среде, что послужило основой для многочисленных курсов по операционным системам и диссертаций.

Операционная система Berkeley UNIX была создана в 1977 г., когда исследовательская группа по вычислительным системам (Computer Systems Research Group, CSRG), организованная в Калифорнийском университете в Беркли, приобрела лицензию на исходный код системы у AT&T. Версии, выпускаемые этой группой, сокращенно назывались BSD (Berkeley Software Distribution). Их выпуск начался с версии 1BSD для компьютера PDP-11 и достиг кульминации в 1993 г., когда вышла версия 4.4BSD.

По мере завоевания системой UNIX коммерческого признания цена лицензий на ее исходный код росла. В конце концов, специалисты университета в Беркли приняли решение убрать код AT&T из BSD. Работа предстояла долгая, утомительная и кропотливая. Незадолго до ее завершения университет лишился финансирования в области исследований операционных систем, и исследовательскую группу по вычислительным системам расформировали.

Перед расформированием группа выпустила окончательную версию системы без кода AT&T под названием 4.4BSD-Lite. Большинство современных версий BSD UNIX (включая FreeBSD, NetBSD Mac OS X[2] и OpenBSD) ведет свое начало именно от этого пакета.

Другие ключевые версии UNIX (включая HP-UX и Solaris) преимущественно являются потомками исходного семейства операционных систем AT&T. Код Linux отличается от кода версий UNIX AT&T или BSD, но с точки зрения функциональных возможностей занимает промежуточное положение между ними.

1.4. Дистрибутивы Linux

🕮 Дополнительные замечания по поводу дистрибутивов приведены в конце этого раздела.

Linux отличается от других вариантов UNIX тем, что, по сути, состоит из одного ядра. Это ядро нужно укомплектовать командами, демонами и другим программным обеспечением, чтобы сформировать полноценную операционную систему, — в терминологии Linux это называется *дистрибутивом*. Все дистрибутивы основаны на едином семействе ядер, однако набор служебных программ, дополняющих ядро, может существенно варьироваться.

Эти "вспомогательные материалы" образованы обширной коллекцией программ, разработанных за последние 30 лет тысячами программистов. Часто утверждают, и отчасти это справедливо, что ссылка на завершенную операционную систему просто как на "Linux" принижает заслуги этих разработчиков и создает искаженное представление об исторических условиях, в которых они работали. К сожалению, альтернативная версия, которую наиболее часто предлагают — "GNU/Linux" — обременена собственным политическим багажом и официально поддерживается только дистрибутивом Debian. В статье "Дискуссия по поводу наименования GNU/Linux" на сайте Wikipedia приведены аргументы обеих сторон.

[2] Строго говоря, ядро Mac OS X — вариант Mach, гибридной системы, включающей в себя как элементы BSD, так и части, которые весьма отличаются от вариантов UNIX.

Дистрибутивы различаются по своему назначению, наличию служб поддержки и степени популярности. В табл. 1.1 перечислены наиболее популярные дистрибутивы общего назначения. Они перечислены в алфавитном порядке, а не по степени популярности.

Таблица 1.1. Наиболее популярные дистрибутивы Linux общего назначения

Дистрибутив	Web-сайт	Комментарии
CentOS	`www.centos.org`	Бесплатный аналог Red Hat Enterprise Linux
Debian	`www.debian.org`	Популярный некоммерческий дистрибутив
Fedora	`fedora.redhat.com`	Вариант Red Hat Linux, предназначенный для использования отдельными пользователями
Gentoo	`www.gentoo.org`	Дистрибутив, ориентированный на использование исходного кода
Mandriva[a]	`www.mandriva.com`	Один из наиболее дружественных по отношению к пользователю дистрибутивов
openSUSE	`www.opensuse.org`	Бесплатный аналог SUSE Enterprise Linux
Red Hat Enterprise	`www.redhat.com`	Дистрибутив Red Hat Linux, ориентированный на использование в крупномасштабных корпоративных системах.
Slackware	`www.slackware.com`	Стабильный базовый, лишенный каких-либо дополнительных возможностей дистрибутив
SUSE Linux Enterprise	`www.novell.com/linux`	Многоязыковой дистрибутив, особенно популярен в Европе
TurboLinux	`www.turbolinux.com`	Многоязыковой дистрибутив, особенно популярен в Азии
Ubuntu	`www.ubuntu.com`	Доработанная версия Debian

[a] Ранее назывался Mandrakelinux

В табл. 1.1 упомянуты далеко не все дистрибутивы, к тому же есть дистрибутивы специального назначения (например, для разработчиков встроенных систем).

Один из представляющих интерес дистрибутивов, который не представлен в табл. 1 — Knoppix (`www.knoppix.com`). Эта версия Linux распространяется на загрузочных компакт-дисках. Ее основное достоинство — возможность использования в качестве восстановительного компакт-диска системы Linux, которая из-за нарушения системы безопасности или вследствие технической проблемы стала недоступной для загрузки. Концепция использования загрузочных компакт-дисков оказалась столь популярной, что в ходе разработка большинства наиболее популярных дистрибутивов предпринимаются действия по ее применению. Теперь, когда система Ubuntu позволяет осуществлять загрузку с дистрибутивного компакт-диска, Knoppix становится менее популярной. Обновляемый перечень дистрибутивов Linux, допускающих загрузку с компакт диска, можно найти по адресу `www.frozentech.com/content/livecd.php`.

В течение более десяти последних лет Red Hat занимает ведущее положение среди вариантов Linux, и его дистрибутивы наиболее популярны в Северной Америке. В 2003 г. первоначальный дистрибутив Red Hat Linux был разделен на серию версий, ориентированных на производственные среды, которые по-

лучили название Red Hat Enterprise Linux (в этой книге мы иногда употребляем аббревиатуру RHEL), и на версии, разрабатываемые в рамках проекта с привлечением всех членов сообщества пользователей и разработчиков, который получил название Fedora. Это разделение было обусловлено рядом технических, экономических, логических и юридических причин, но пока оба эти дистрибутива остаются схожими. RHEL отличается высоким уровнем поддержки и стабильностью, но его по существу невозможно использовать, не приобретя лицензию в компании Red Hat.

Проект CentOS (`www.centos.org`) концентрирует исходный код, который компания Red Hat обязана распространять в соответствии с различными лицензионными соглашениями (наиболее значимой из них является общедоступная лицензия GNU), и собирает их в законченный дистрибутив, который во многом подобен дистрибутиву Red Hat Enterprise Linux, но доступен бесплатно. Этот дистрибутив лишен торговой марки компании Red Hat и не содержит некоторых запатентованных программных средств, но в остальных отношениях аналогичен платной версии. CentOS стремится к полной совместимости с RHEL, как по двоичному коду, так и по степени исправления ошибок.

Этот дистрибутив — прекрасный выбор для тех сайтов, которые стремятся развернуть крупномасштабную производственную систему, не платя при этом "десятину" компании Red Hat. Возможен также смешанный подход: сетевые серверы могут работать под управлением Red Hat Enterprise Linux, используя при этом преимущества, предоставляемые прекрасным уровнем поддержки со стороны Red Hat, а рабочие станции могут функционировать под управлением CentOS. Такой подход обеспечивает оптимизацию с точки зрения риска и поддержка, в то же время сводя к минимуму затраты и сложность администрирования.

Компания SUSE, которая теперь является одним из подразделений корпорации Novell, недавно пошла по пути Red Hat и начала распространять два связанных дистрибутива: openSUSE, который содержит только бесплатное программное обеспечение, и платный SUSE Linux Enterprise, который включает средства формальной поддержки и предоставляет несколько дополнительных возможностей. В прошлом предпринимались попытки скрыть существование бесплатной версии SUSE, но руководство Novell более откровенно в отношении этого издания, чем предыдущие владельцы SUSE. Теперь наиболее свежую информацию можно получить непосредственно на сайте `www.opensuse.org`. Эта книга не содержит никакой информации, характерной для того или иного дистрибутива SUSE, поэтому мы называем их всех просто "SUSE".

Дистрибутивы Debian и Ubuntu сохраняют идеологическую направленность на разработку членами сообщества пользователей и разработчиков и открытый доступ, поэтому вопрос о том, какие части дистрибутива бесплатны или разрешены для дальнейшего распространения, даже не возникает. Debian обязан своим существованием стараниям и доброй воле сообщества GNU, а Ubuntu в настоящее время финансируется за счет благотворительных пожертвований южноафриканского предпринимателя Мака Шаттлеворта. Ubuntu даже готова высылать по почте бесплатные компакт-диски без оплаты почтовых услуг.

Какой дистрибутив наилучший

Как показывает опыт, вынесенный в заголовок вопрос едва ли не чаще всего задается в Linux-форумах, и редко можно встретить ответ на него. На самом деле ответ зависит от того, как именно вы собираетесь использовать систему, с какими версиями UNIX знакомы, каковы ваши симпатии и на какую поддержку вы рассчитываете.

Большинство дистрибутивов позволяет делать все, что может потребоваться от Linux-систем. В некоторых из них нужно инсталлировать дополнительное программное обеспечение, чтобы система стала полнофункциональной, другие ускоряют выполнение определенных задач, но в целом между ними нет существенных различий. Остается загадкой: зачем столько дистрибутивов? Причем отличительными свойствами каждого из них являются "простота инсталляции" и "внушительная библиотека программных средств". Нет-нет, да и проскакивает мысль о том, что кто-то просто испытывает удовлетворение от выпуска новых дистрибутивов Linux.

Поскольку эта книга сфокусирована на управлении крупномасштабными Linux-системами, мы больше склоняемся к дистрибутивам наподобие Red Hat Enterprise Linux, которые предусматривают управление сетями. Одни дистрибутивы создавались в расчете на промышленное применение, другие — нет. Какой-нибудь дополнительный программный компонент системы промышленного уровня, на первый взгляд малозначительный, в действительности существенно упрощает администрирование.

Принимая решение в пользу того или иного дистрибутива, обращайте внимание не только на его функциональные возможности, но и на то, как будет протекать сотрудничество вашей организации и поставщика дистрибутива в ближайшие годы. Задайте себе ряд важных вопросов.

- Будет ли дистрибутив существовать в ближайшие пять лет?
- Будут ли оперативно устраняться бреши в системе защиты?
- Будут ли оперативно выпускаться новые версии программных продуктов?
- Будет ли оказана помощь в решении возникших проблем?

Под этим углом зрения некоторые чрезвычайно интересные, компактные дистрибутивы уже не кажутся столь привлекательными. С другой стороны, наиболее жизнеспособные дистрибутивы не обязательно имеют корпоративный статус. Например, мы ожидаем, что система Debian (простите, Debian GNU/Linux!) просуществует довольно долго, несмотря на то что Debian — это не коммерческая компания, она ничего не продает и не предлагает сервисного обслуживания.

Обширный перечень дистрибутивов, в том числе не англоязычных, можно найти по адресам `www.linux.org/dist`, `lwn.net/Distributions` или `distrowatch.com`.

В этой книге мы остановились на пяти популярных дистрибутивах: Red Hat Enterprise Linux 4.3 ES, Fedora Core 5, SUSE Linux Enterprise 10.2, Ubuntu 6.06 ("Dapper Drake"), и на тестируемой в настоящее время (на сентябрь 2006 г. версии Debian GNU/Linux 3.2 ("Etch"). Они представляют собой срез рынка корпоративных Linux-систем и охватывают большинство инсталляций Linux в крупных организациях.

Административные средства конкретных дистрибутивов

Во многих дистрибутивах имеются графические средства (такие как Red Hat Network Administration Tool и YaST2 в SUSE), упрощающие конфигурирование и администрирование определенных компонентов системы. Эти средства очень удобны, особенно ад-

министраторам-новичкам, но они скрывают детали того, что на самом деле происходит в системе при внесении изменений. В этой книге мы описываем низкоуровневые механизмы, а не высокоуровневые инструментальные средства.

На то есть ряд причин. Во-первых, графические средства зачастую являются платными или, по крайней мере, зависят от дистрибутива. В результате то, что на низком уровне является одинаковым для всех дистрибутивов, начинает казаться хаотичным и непостоянным. Во-вторых, мы считаем, что для системного администратора важно четко знать, как работает система. Когда в системе происходит сбой, графические утилиты не всегда помогают найти и устранить проблему.

Наконец, конфигурировать систему на низком уровне обычно удобнее: так быстрее и надежнее, больше возможностей для манипуляции, легче писать сценарии.

1.5. Типографские особенности книги

Имена файлов, команды и аргументы команд, которые следует набирать на клавиатуре без изменений, даны моноширинным шрифтом. Аргументы, вместо которых следует подставлять конкретные значения, выделены курсивом. Например, в команде

```
cp файл каталог
```

предполагается, что аргумент *файл* следует заменить именем реального файла, а аргумент *каталог* — именем реального каталога.

Результаты работы команд, а также фрагменты сценариев и файлов конфигурации даны моноширинным шрифтом. Комментарии к интерактивным сеансам иногда даются курсивом, например:

```
$ grep Bob /pub/phonelist          /* Найти номер телефона Боба */
Bob Knowles 555-2834
Bob Smith 555-2311
```

При описании синтаксиса команд мы, как правило, используем те же обозначения, что и в интерактивном руководстве:

- текст, заключенный в квадратные скобки (“[” и “]”), является необязательным;
- текст, после которого стоит многоточие (“...”), можно повторять;
- фигурные скобки (“{” и “}”) указывают на то, что необходимо выбрать один из элементов, разделенных вертикальной чертой (“|”).

Например, спецификация

```
bork [-x] {on|off} имя_файла ...
```

соответствовала бы любой из следующих команд:

```
bork on /etc/passwd
bork -x off /etc/passwd /etc/termcap
bork off /usr/lib/tmac
```

В выражениях с шаблонами поиска используются следующие метасимволы:

- звездочка (*) обозначает нуль или более символов;
- знак вопроса (?) обозначает один символ;
- тильда (~) обозначает начальный каталог текущего пользователя;

- выражение *~пользователь* обозначает начальный каталог указанного пользователя.

Например, иногда мы обозначаем каталоги, где хранятся сценарии запуска Debian (`/etc/rc0.d`, `/etc/rc1.d` и т.д.), сокращенным шаблоном `/etc/rc*.d`.

Информация по конкретным дистрибутивам

Приведенная в книге информация относится, как правило, ко всем вышеупомянутым дистрибутивам, если нет специальных уточнений. Подробности, касающиеся конкретного дистрибутива, помечаются значком поставщика.

Red Hat® Enterprise Linux® 4.3 ES

Fedora™ Core 5

SUSE® Linux Enterprise 10.2

Ubuntu® 6.06 “Dapper Drake”

Debian® GNU/Linux 3.2 “Etch” (тестовая версия 9/06)

Эти значки использованы с разрешения их владельцев. Следует заметить, что разработчики приведенных дистрибутивов не рецензировали эту книгу.

1.6. Где искать информацию

Документация к Linux распределена по целому ряду источников, часть из которых находится в самой системе, а часть — в сети Internet. Перечислим основные из них:

- страницы интерактивного руководства, или man-страницы, предназначенные для просмотра с помощью команды `man`;
- документы Texinfo, предназначенные для просмотра с помощью команды `info`;
- документы типа HOWTO — короткие заметки по разным темам (`www.tdlp.org`);
- руководства — более объемные материалы по разным темам (`www.tdlp.org`);
- документация по конкретным дистрибутивам;
- Web-страницы, посвященные конкретным программным проектам.

Страницы интерактивного руководства и документы Texinfo составляют традиционную “интерактивную” документацию (хотя, конечно, вся вышеперечисленная документация в той или иной мере является интерактивной).

Эти материалы в основном инсталлируются вместе с системой. Исключение составляют man-страницы отдельных программ, инсталлируемые вместе с соответствующим пакетом.

Страницы интерактивного руководства содержат полное описание отдельных команд, драйверов, форматов файлов и библиотечных функций. В них не найти ответа на общие вопросы, например “Как инсталлировать новое устройство?” или “Почему моя

система работает так медленно?". Ответы на такие вопросы следует искать в документах HOWTO.

Система Texinfo была изобретена энтузиастами проекта GNU после того, как выяснилось, что команда `nroff`, применяемая для форматирования man-страниц, является собственностью AT&T. Теперь то же самое делает GNU-команда `groff`, так что проблема утратила актуальность, но, к сожалению, многие GNU-пакеты требуют, чтобы документация к ним была в представлена формате Texinfo. Таким образом, появляется дополнительный, совершенно ненужный стандарт документации. Кроме того, Texinfo — довольно запутанная гипертекстовая система.

Чтобы избежать сложностей, связанных с Texinfo, посредством команды `less` перенаправьте поток вывода команды `info`, что позволит обойти ее встроенную навигационную систему. В качестве побочного эффекта вы получите также возможность использования встроенных поисковых средств команды `less`.

Но есть и приятный момент. Пакеты, документируемые с помощью системы Texinfo, обычно инсталлируют страницы-заглушки, в которых сообщается о том, что информацию о соответствующем пакете можно получить, вызвав команду `info`.

Таким образом, можно продолжать пользоваться командой man, обращаясь к команде `info` лишь в случае необходимости. Команда `info info` выдает справку по запутанной системе Texinfo.

Документы типа HOWTO и руководства находятся в ведении проекта Linux Documentation Project, сокращенно LDP. Эти документы доступны на Web-сайте `www.tldp.org`. Он представляет собой центральное хранилище разного рода информации, посвященной Linux. Здесь же концентрируются результаты усилий по переводу документации на различные языки.

Некоторые общедоступные руководства представляют особый интерес для системных администраторов. Среди них: *The Linux System Administrators' Guide*, *Advanced Bash-Scripting Guide*, *The Linux Network Administrator's Guide (Second Edition)* и *Linux System Administration Made Easy*.

К сожалению, многие LDP-документы практически не обновляются, поэтому быстро устаревают. Всегда обращайте внимание на дату публикации документа, так как по ней можно судить об актуальности информации.

Ряд важнейших компонентов Linux поддерживается сторонними организациями, такими как ISC (Internet Software Consortium — консорциум разработчиков программного обеспечения для Internet) и ASF (Apache Software Foundation — организация разработчиков программного обеспечения для Apache). Эти организации обычно предоставляют и документацию к распространяемым пакетам. Некоторые поставщики включают в дистрибутив тот или иной программный пакет без документации, поэтому в таких случаях необходимо интересоваться, имеются ли дополнительные материалы.

Другим ценным источником информации о программных пакетах Linux является серия документов RFC (Request for comments — запрос на получение комментариев), в которых описываются протоколы и программное обеспечение сети Internet (подробности можно найти в разделе 12.1).

Организация страниц интерактивного руководства

В Linux man-страницы обычно делятся на девять разделов (табл. 1.2).

Таблица 1.2. Разделы man-страниц Linux

Раздел	Содержание
1	Команды пользовательского уровня и приложения
2	Системные вызовы и коды ошибок ядра
3	Библиотечные функции
4	Драйверы устройств и сетевые протоколы
5	Стандартные форматы файлов
6	Игры и демонстрационные программы
7	Различные файлы и документы
8	Команды системного администрирования
9	Внутренние интерфейсы и спецификации ядра

Некоторые разделы делятся на подразделы. Например, подраздел 3M содержит страницы с информацией о библиотеке математических функций системы. Разделы 6 и 9 обычно пусты. Многие системы имеют раздел руководства под названием “l” для локальных man-страниц. Для описания внутренних команд отдельных программ (например, команд интерпретатора bash) общепринятым является раздел “n”.

Неформатированная информация для man-страниц (входные данные команды `nroff`) обычно хранится в каталогах `/usr/share/man/man`*X*, где *X* — цифра от `1` до `9` либо буква ‘`l`’ или ‘`n`’. В целях экономии места на диске страницы сжимаются с помощью утилиты gzip (команда man может очень быстро разархивировать их). Команда man форматирует страницы непосредственно в процессе отображения.

Форматированные версии страниц находятся в каталоге `/var/cache/man/cat`*X*. Команда `man` форматирует страницы непосредственно в процессе отображения. Если каталоги **cat***X* доступны для записи, эта команда сохраняет отформатированные страницы по мере их создания, формируя кэш наиболее часто читаемых страниц.

Команда `man` ищет страницы в целом ряде каталогов. Определить путь поиска позволяет команда `manpath`. Результат ее работы (в Fedora) обычно таков:

```
$ manpath
/usr/kerberos/man:/usr/local/share/man:/usr/share/man/en:/usr/share/man
```

Эта установка хранится в переменной среды `MANPATH`, и в случае необходимости ее можно изменить. Соответствующий общесистемный параметр задан в файле `/etc/man.config` (RHEL и Fedora) или `/etc/manpath.config` (SUSE, Debian и Ubuntu).

Команда `man`: чтение страниц интерактивного руководства

Команда `man заголовок` форматирует конкретную страницу интерактивного руководства и выводит ее на терминал пользователя посредством программы `less` (или другой программы постраничной разбивки, которая задана в переменной среды `PAGER`). Аргумент `заголовок` — это, как правило, имя команды, устройства или файла, о которых необходимо получить справочную информацию. Поиск по разделам руководства осуществляется в порядке возрастания номеров, хотя разделы, посвященные описанию команд (1, 8 и 6), обычно просматриваются в первую очередь.

Команда `man раздел заголовок` вызывает man-страницу из указанного раздела. Так, команда `man tty` отображает справку по команде `tty`, а команда `man 4 tty` — страницу для драйвера управляющего терминала.

Команда `man -k` *ключевое_слово* выводит список man-страниц, в строке пояснений к которым имеется указанное ключевое слово.

Например:

```
$ man -k translate
objcopy (1)          - copy and translate object files
dcgettext (3)        - translate message
tr (1)               - translate or delete characters
snmptranslate (1)    - translate SNMP OID values into more useful
information
tr (1p)              - translate characters
gettext (1)          - translate message
ngettext (1)         - translate message and choose plural form ...
```

Другие источники информации по Linux

Невозможно перечислить все полезные источники информации по Linux, поэтому упомянем лишь важнейшие из них (табл. 1.3).

Таблица 1.3. Web-ресурсы, посвященные Linux

Web-сайт	Описание
`linux.slashdot.org`	Огромный архив новейшей справочной технической информации по Linux
`lwn.net`	Сборник материалов по Linux и программам с открытым исходным кодом
`www.freshmeat.net`	Большой каталог программ для Linux и UNIX
`www.kernel.org`	Официальный Web-сайт разработчиков ядра Linux
`www.linux.com`	Неофициальное хранилище информации по Linux
`www.linux.org`	Еще одно неофициальное хранилище информации по Linux
`www.linuxhq.com`	Хранилище информации по ядру и “заплат” к нему
`www.linuxworld.com`	Интерактивный журнал от авторов ComputerWorld
`www.tldp.org`	Проект LDP (Linux Documentation Project)
`www.tucows.com`	Мультисистемный архив программного обеспечения

Не бойтесь обращаться к ресурсам, посвященным UNIX: большая часть информации в них напрямую применима к Linux. Информация по вопросам системного администрирования доступна в сети во множестве форм. К примеру, вопросы, касающиеся системного администрирования, можно вводить в таких поисковых системах, как Google, Yahoo! или Ask. Список ресурсов, к которым могут обращаться начинающие администраторы, приведен в главе 30.

Многие Web-сайты непосредственно посвящены проблемам системного администрирования. Вот некоторые из них:

- `www.ugu.com` — аббревиатура “ugu” расшифровывается как “UNIX Guru Universe”, т.е. “Вселенная гуру UNIX”; на этом сайте содержится много информации для системных администраторов;

- www.stokely.com — обширная коллекция ссылок на ресурсы, связанные с системным администрированием;
- www.tucows.com — качественное программное обеспечение;
- slashdot.org — место, где публикуются различные оригинальные новости;
- www.cpan.org — центральный архив сценариев и библиотек Perl;
- securityfocus.com — Web-сайт, посвященный вопросам безопасности, огромная поисковая база данных.

Другим забавным и полезным источником является страница "Rosetta Stone" (розеттский камень) по адресу bhami.com/rosetta.html. Здесь рассказывается о командах и утилитах, применяемых для решения основных административных задач в различных операционных системах.

1.7. Как инсталлировать программное обеспечение

В дистрибутивах Linux программное обеспечение разделено на пакеты, которые можно инсталлировать независимо друг от друга. При установке системы на новый компьютер пользователь выбирает группу "стартовых" пакетов, копируемых в систему.

Подобный подход упрощает многие аспекты системного конфигурирования и является одним из ключевых преимуществ Linux в сравнении с традиционными разновидностями UNIX. К сожалению, он же усложняет задачу описания дистрибутивов, поскольку не всегда ясно, какие пакеты являются "частью" того или иного дистрибутива. Можно ли считать таковым пакет, если он есть на инсталляционном компакт-диске, но не устанавливается по умолчанию? А если он имеется на дополнительных компакт-дисках, которые прилагаются к "профессиональной" версии дистрибутива?

В этой книге мы в основном описываем стандартный вариант инсталляции демонстрационных дистрибутивов. Если говорится о том, что такой-то пакет отсутствует в стандартном варианте инсталляции, то это не означает, что его вообще нет в системе или он не поддерживается в данном дистрибутиве. Ниже рассказывается о том, как узнать, что входит в дистрибутив, и как достать то, чего в нем нет.

В первую очередь воспользуйтесь командой which, чтобы определить, "прописана" ли нужная утилита в строке поиска (переменная среды PATH). Например, следующая команда сообщает о том, что компилятор GNU C инсталлирован в каталоге /usr/sbin:

```
$ which gcc
/usr/bin/gcc
```

Если команда which не помогла, воспользуйтесь командой whereis, которая ведет поиск в системных каталогах без учета строки поиска. Учтите также, что в некоторых системах команда which не отображает файлы, для которых отсутствует право выполнения. Например:

```
$ which ipppd
/usr/bin/which: no ipppd in (/bin:/usr/bin:/sbin:/usr/sbin)
$ whereis ipppd
ipppd: /usr/sbin/ipppd
$ ls -l /usr/sbin/ipppd
-rwx------   1 root   root     124924 Aug  3  2000 /usr/sbin/ipppd
```

Другой вариант — это чрезвычайно удобная команда `locate`, которая просматривает предварительно скомпилированный индекс файловой системы в поиске имен файлов, соответствующих заданному шаблону.

С ее помощью можно искать не только программы или пакеты, но и файлы любых типов. Например, если не известно точно, где искать файл заголовков `signal.h` (в нем содержатся определения сигналов Linux), попробуйте поступить так:

```
$ locate signal.h
/usr/include/asm/signal.h
/usr/include/linux/signal.h
/usr/include/signal.h
/usr/include/sys/signal.h
```

База данных команды `locate` обычно регенерируется каждую ночь командой `updatedb`, которая запускается демоном `cron`. Следовательно, результаты работы команды `locate` не всегда отражают последние изменения в системе.

Если известно имя искомого пакета, воспользуйтесь системными утилитами обработки пакетов, которые непосредственно проверят наличие пакета. Например, в Red Hat, Fedora и SUSE следующая команда определяет наличие интерпретатора языка Python:

```
$ rpm -q python
python-1.5.2-27
```

📖 Подробнее об утилитах работы с пакетами рассказывается в главе 11.

Если нужный пакет не инсталлирован, прежде всего, его следует поискать с помощью системы автоматического управления пакетами данного дистрибутива. Все дистрибутивы поддерживают ту или иную форму системы обновления старых и поиска новых пакетов в Internet. Наиболее распространенными системами являются `yum` и APT, описанные в разделе 11.4.

Например, в системе Debian, в которой использована APT, для получения и установки последней версии Python можно использовать следующую команду:

```
# apt-get install python
```

Программное обеспечение для Linux создается независимыми группами разработчиков, выпускающих программы в виде исходных кодов. Авторы дистрибутивов затем берут исходные коды, компилируют их в соответствии с особенностями конкретной системы и включают в дистрибутив полученные бинарные файлы. Как правило, намного проще инсталлировать бинарную версию пакета, предназначенную для данного дистрибутива, чем искать и компилировать исходный код. Но нужно учитывать, что иногда авторы дистрибутивов на один-два выпуска отстают от текущей версии пакета.

Если в двух дистрибутивах используется одна и та же система обработки пакетов, то это еще не означает, что пакеты обоих дистрибутивов будут взаимозаменяемыми. Например, в Red Hat и SUSE применяется система RPM, однако структура их файловых систем неодинакова. Рекомендуется всегда пользоваться пакетами, созданными для конкретного дистрибутива.

Если поиск не дал результатов, посетите какой-нибудь Web-архив программ, например `freshmeat.net`, или просто введите имя пакета в поисковой системе Google.

1.8. Основные задачи системного администратора

В этом разделе кратко перечислены задачи, обычно возлагаемые на системного администратора. Совсем не обязательно, чтобы эти функции выполнял один человек. Во многих организациях работа распределяется между несколькими администраторами. В любом случае необходим хотя бы один человек, который понимал бы все поставленные задачи и обеспечивал их выполнение другими людьми.

Добавление, удаление и управление учетными записями пользователей

📖 Подробнее о добавлении новых пользователей рассказывается в главе 6.

Обязанностью системного администратора является создание учетных записей для новых пользователей и удаление учетных записей тех пользователей, которые уже не работают в системе. Процесс управления записями можно автоматизировать, но ряд решений, связанных с включением в систему нового пользователя (место размещения его начального каталога, компьютер, на котором должна создаваться учетная запись, и т.д.), должен принимать только администратор.

Если необходимо прекратить доступ пользователя к системе, его учетная запись должна быть аннулирована. Все файлы, относящиеся к этому пользователю, необходимо удалить, чтобы они не занимали место на диске.

Подключение и удаление аппаратных средств

📖 Дополнительная информация по данной теме приведена в главах 7, 28 и 23.

В случае приобретения новых аппаратных средств или подключения уже имеющихся устройств к другому компьютеру нужно переконфигурировать систему таким образом, чтобы она распознала и активизировала эти устройства. Изменение конфигурации может быть как простой задачей (например, подключение принтера), так и более сложной (например, подключение жесткого диска).

Резервное копирование

📖 Подробнее о резервном копировании рассказывается в главе 9.

Резервное копирование является, наверное, одной из наиболее важных задач системного администратора, которая, к сожалению, чаще всего игнорируется или выполняется спустя рукава. Процедура резервного копирования довольно утомительна и занимает много времени, но осуществлять ее необходимо. Этот процесс можно автоматизировать или поручить подчиненным, но все равно системный администратор обязан убедиться в том, что резервное копирование выполнено правильно и по графику (а с полученного носителя можно восстановить данные).

Инсталляция и обновление программ

📖 Подробнее об утилитах работы с программами рассказывается в главе 11.

После приобретения нового программного обеспечения его нужно инсталлировать и протестировать, часто в нескольких операционных системах и на различном оборудова-

нии. Если программы работают нормально, пользователям необходимо сообщить об их наличии и местонахождении. Выпускаемые пакеты обновлений для исправления ошибок и устранения брешей в системе безопасности должны без проблем устанавливаться в локальных системах.

Локальное программное обеспечение следует инсталлировать туда, где его можно будет легко отличить от системных программных средств. Это значительно упрощает задачу обновления операционной системы, поскольку исчезает опасность уничтожения локальных программ в ходе подобного обновления.

Мониторинг системы

Крупные системы требуют неусыпного контроля. Существует множество обязательных ежедневных операций, а именно: проверка правильности функционирования электронной почты и Web-сервера; просмотр журнальных файлов на предмет наличия ранних признаков неисправностей; контроль над подключением локальных сетей; контроль наличия системных ресурсов (в частности, проверка наличия свободного места на диске).

Поиск неисправностей

Linux-системы и аппаратные устройства, на которых они работают, время от времени выходят из строя. Задача администратора — диагностировать сбои в системе и в случае необходимости вызывать специалистов. Как правило, найти неисправность бывает намного сложнее, чем устранить ее.

Ведение локальной документации

📖 Рекомендации, касающиеся ведения документации, даны в разделе 29.9.

При настройке конфигурации системы под конкретные требования очень скоро обнаруживается, что она значительно отличается от базовой конфигурации, которая описана в документации. Поэтому системный администратор должен документировать все особенности локальной конфигурации, в том числе инсталлируемые программные средства, не входящие в стандартный комплект поставки, и разводку сетевых кабелей.

Администратор должен также вести учет обслуживания всех аппаратных средств, регистрировать состояние резервных копий, документировать локальные правила работы в системе.

Пристальное слежение за системой безопасности

📖 Вопросы безопасности рассматриваются в главе 20.

Системный администратор отвечает за реализацию стратегии защиты и должен периодически проверять, не нарушена ли безопасность системы. В системах с низким уровнем безопасности эта процедура может быть сведена к нескольким текущим проверкам на предмет несанкционированного доступа.

В системах с высоким уровнем безопасности обычно применяется сложная сеть ловушек и программ контроля.

Оказание помощи пользователям

О необходимости оказания помощи пользователям в решении различных проблем редко упоминается в должностной инструкции системного администратора, хотя выполнение подобного рода обязанностей "съедает" бо́льшую часть его рабочего времени. Системных администраторов терроризируют самыми разными вопросами, начиная от "Вчера моя программа работала, а сегодня нет! Что вы поменяли?" до "Я пролила кофе на клавиатуру! Нужно ли теперь полить ее водой, чтобы смыть кофе?".

1.9. Издержки профессии

Системные администраторы — настоящие "многостаночники". Они часто имеют другую работу, просто их попросили присмотреть за несколькими компьютерами "на стороне". Если вы один из таких людей, подумайте, к чему в конце концов это может привести.

Чем больше вы будете знать о своей системе, тем больше пользователи будут зависеть от вас. Сети неуклонно разрастаются, и вы будете вынуждены тратить все больше и больше времени на выполнение функций администратора. Вскоре окажется, что вы — единственный человек во всей организации, который знает, как решить целый ряд важнейших проблем.

Если коллеги стали считать вас локальным системным администратором, от этой роли уже трудно отказаться. Мы знаем нескольких людей, которым пришлось поменять место работы, чтобы избавиться от дополнительной нагрузки. Поскольку круг обязанностей системного администратора четко очертить нельзя, от вас, скорее всего, потребуют, чтобы вы были не только штатным администратором, но и штатным инженером, писателем, а также секретарем.

Чтобы избежать потока просьб о помощи, некоторые администраторы начинают плохо выполнять свои обязанности, становятся раздражительными. Не стоит придерживаться такого подхода, так как вы будете плохо выглядеть в глазах окружающих, и у вас могут возникнуть дополнительные проблемы.

Вместо этого мы предлагаем просто фиксировать время, затрачиваемое на системное администрирование.

Ведите работу на должном уровне и собирайте доказательства, которые можно будет использовать, когда вы попросите начальника освободить вас от обязанностей администратора. В большинстве организаций для того, чтобы добиться замены, приходится упрашивать руководство полгода, а то и год, так что учитывайте это в своих планах.

С другой стороны, может оказаться, что системное администрирование вам нравится, и вы захотите стать штатным администратором. В этом случае проблем с поиском работы у вас не будет.

К сожалению, сама работа от этого не станет легче. О политических аспектах системного администрирования рассказывается в главе 30.

Синдром хронического администрирования

Малоприятной, но, увы, распространенной болезнью системных администраторов является синдром хронического администрирования. Признаки заболевания обычно проявляются на третий год после начала работы администратором и могут привести к

преждевременному выходу на пенсию. Ниже приведен неполный список характерных симптомов.

- Острая пейджерофобия — раздражающее ощущение того, что у вас сработал пейджер и мирный вечер с супругой придется прервать. Вам кажется, что вас срочно вызывают для устранения последствий ЧП и вам предстоит работать 72 часа подряд без перерывов на еду.
- Навязчивая пользователемания — маниакальное стремление протыкать иголками восковые фигурки отдельных представителей пользовательского племени, которые не понимают, что проблемы с администрированием возникают из-за отсутствия грамотного планирования с их стороны.
- Идиопатическая лентоплексия — внезапно проявляющееся поздней ночью острое желание смонтировать ленточный накопитель для резервного копирования и убедиться в том, что он читается и маркирован правильно.
- Интеллектуальная шизоидная нетерпимость — непреодолимое желание стукнуть знакомого системного администратора, который никогда не слышал о научных методах администрирования.

Для лечения болезни могут использоваться различные терапевтические средства. Наиболее эффективными являются принудительное развитие чувства юмора и организация небольшого, но хорошо оборудованного винного погребка в офисе. Допускаются также более медитативные методы, например, молчаливо-безучастное разглядывание окружающего пространства, когда рядом раздается гневное “Что? Сервер снова упал?!”. Если ничего не помогает, возьмите отпуск.

1.10. Рекомендуемая литература

Наилучший печатный источник информации, полезной системным администраторам (кроме книги, которую вы держите в руках) — серия книг, выпускаемая издательством O’Reilly. Начало ей положила книга *Linux in a Nutshell*, изданная более 20 лет назад, и теперь в ней издаются отдельные тома практически по каждой из наиболее важных подсистем и команд UNIX и Linux. Выпускаются также серии книг, посвященных Internet, Windows и другим темам, не связанным с UNIX. Все книги имеют доступные цены, издаются своевременно и посвящены вполне определенным темам. Тим О’Рейли является достаточно активным приверженцем движения по распространению программного обеспечения с открытым исходным кодом и регулярно проводит посвященную этому конференцию, OSCON, а также конференции по другим сложным техническим вопросам. Конференция OSCON проводится дважды в год — один раз в США, другой в Европе. Более подробную информацию можно найти по адресу `www.oreilly.com`.

Хотя существует не один десяток книг по Linux, нам пока не встретилась такая, которую можно было бы безоговорочно рекомендовать в качестве пособия. Лучше обратиться к “классике” UNIX. Практически все, что вы там прочтете, в равной степени применимо и в Linux.

- Siever, Eellen, Aaron Weber, and Stephen Figgins. *Linux in a Nutshell (5th Edition)*. Sebastopol, CA: O’Reilly Media, 2006.
- Lamb, Linda, and Arnold Robbins. *Learning the vi Editor (6th Edition)*. Sebastopol, CA: O’Reilly & Associates, 1998.

- Powers, Shelly, Jerry Peek, Tim O'Reilly, and Mike Lourkides. *UNIX Power Tools (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.
- Wall, Larry, Tom Christiansen, and Jon Orwant. *Programming Perl (3rd Edition)*. Cambridge, MA: O'Reilly Media, 2000.
- Christiansen, Tom, and Nathan Torkington. *Perl Cookbook (2nd Edition)*. Sebastopol, CA: O'Reilly Media, 2003.
- Gancarz, Mike. *Linux and the Unix Philosophy*. Boston: Digital Press, 2003.
- Salus, Peter. *The Daemon, the GNU & the Penguin*. Groklaw. 2006.

В настоящее время эта захватывающая история развития программного обеспечения с открытым исходным кодом, исследованная наиболее известным историком UNIX, переводится в цифровую форму на сайте `groklaw.com` в соответствии с лицензией Creative Commons (Сообщество представителей творческих профессий). Работа уже выполнена на три четверти. URL-адрес самой книги достаточно длинен; попытайтесь найти действующую ссылка на сайте `groklaw.com` или проверьте следующий сокращенный вариант: `tinyurl.com/d6u7j`.

1.11. Упражнения

1.1. Какую команду нужно ввести, чтобы получить информацию о системном вызове `sync` (а не команде `sync`)? Как прочитать соответствующую страницу документации, хранящуюся в каталоге `/usr/local/share/man`?

1.2. Определяет ли общий конфигурационный файл поведение команды `man` в вашей системе? Какие строки нужно в него добавить, чтобы локальные материалы хранилась в каталоге `/doc/man`? Как организовать каталог `/doc/man`, чтобы его структура соответствовала иерархии man-страниц?

1.3. Каковы основные различия между командами `man` и `info`? Каковы преимущества каждой из них?

☆ 1.4. Как в настоящее время протекает разработка ядра Linux? Каковы наиболее актуальные вопросы? Какие организации играют в этом процессе ключевую роль? Кто руководит проектом?

☆ 1.5. Изучите возможности нескольких дистрибутивов Linux (см. раздел 1.4) и порекомендуйте дистрибутив для каждого из нижеперечисленных случаев. Аргументируйте свой выбор.

а) Домашний компьютер пользователя.

б) Университетская компьютерная лаборатория.

в) Корпоративный Web-сервер.

☆ 1.6. Предположим, вы обнаружили, что определенное свойство демона `httpd` сервера Apache работает не так, как описано в документации к Fedora Core 5.

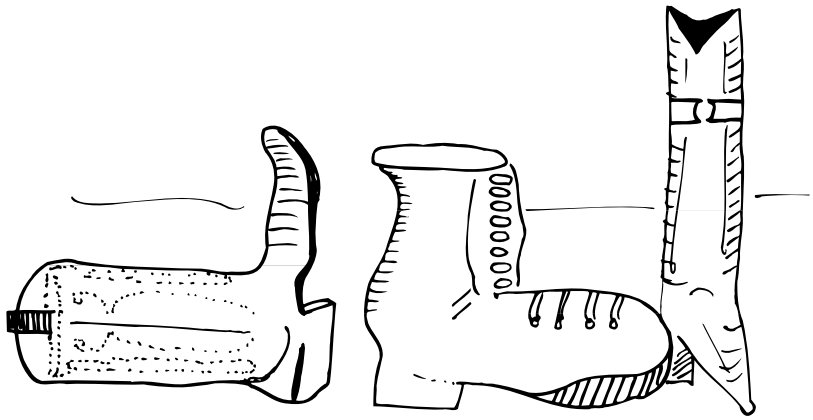
а) Что нужно сделать перед тем, как сообщать об ошибке?

б) Если вы решили, что это действительно ошибка, то кого нужно уведомить и каким способом?

в) Какую информацию необходимо включить в отчет об ошибке?

Глава 2

Запуск и останов системы



Linux — сложная операционная система, и процедура ее включения/выключения не сводится к простому нажатию кнопки питания. Следовательно, чтобы система работала корректно, выполняйте операции запуска и останова по всем правилам.

Хотя процесс начальной загрузки системы всегда был достаточно сложным, все же он был несколько проще в те дни, когда изготовители определяли буквально все аспекты аппаратного и программного обеспечения. Теперь, когда Linux управляет аппаратным обеспечением персональных компьютеров (ПК), процедура загрузки должна выполняться по правилам ПК. При этом приходится иметь дело с множеством возможных конфигураций.

Хотя эта глава в книге — одна из первых, в ней мы иногда оперируем понятиями, которые подробно рассматриваются лишь через несколько сотен страниц. Поэтому рекомендуем также прочесть главы 5, 28 и 29. Если ваша система загружается без проблем, можете пропустить эту главу и вернуться к ней позже.

2.1. Начальная загрузка

Под начальной загрузкой понимается запуск системы при включении питания. Поскольку обычные средства операционной системы на данном этапе еще недоступны, система должна в буквальном смысле “обслужить себя самостоятельно”. В ходе этого процесса ядро системы загружается в память и активизируется. Затем выполняется ряд инициализационных задач, после чего система готова к обслуживанию пользователей.

Начальная загрузка — это период особой уязвимости системы. Ошибки в конфигурационных файлах, сбои в работе или отсутствие нужного оборудования, повреждения

файловых систем могут помешать компьютеру нормально начать работу. Настройка режимов загрузки часто является одной из первых задач, которую приходится решать администратору в новой системе. К несчастью, эта задача — одна из наиболее сложных, и для ее решения необходимо хорошо знать Linux.

Когда включается питание, запускается на выполнение загрузочный код, хранящийся в ПЗУ. Он должен запустить ядро. Ядро опрашивает состояние аппаратных устройств, а затем запускает демон `init`, идентификатор которого всегда равен 1.

Прежде чем на экране появится регистрационное приглашение, должен произойти целый ряд событий. Файловые системы должны быть проверены и смонтированы, а системные демоны — запущены. Соответствующие процедуры реализуются с помощью сценариев интерпретатора команд, которые последовательно запускаются демоном `init`. Эти сценарии часто называют rc-файлами, поскольку они имеют префикс "rc". Он расшифровывается как "run command" (команда запуска) и является пережитком, оставшимся в наследство от операционной системы CTSS, существовавшей примерно в 1965 году. Конкретная структура сценариев и способ их выполнения зависят от системы. Все эти вопросы будут рассмотрены в данной главе.

Автоматическая и ручная загрузка

Linux-системы могут загружаться автоматически либо вручную. В первом случае система запускается самостоятельно, без какого-либо вмешательства извне. Во втором случае также все происходит автоматически до определенного момента, а перед выполнением основных инициализирующих сценариев управление передается оператору (человеку, сидящему за терминалом). В этот момент система входит в так называемый "однопользовательский режим". Большинство системных процессов еще не начало выполняться, и вход других пользователей в систему невозможен.

В повседневной работе почти всегда применяется автоматическая загрузка. Типичная процедура загрузки на современном компьютере выглядит так: пользователь включает питание и ждет (ждет, ждет, ждет...), пока система не перейдет в диалоговый режим. Системный администратор, однако, обязан не только понимать, как проходит автоматическая загрузка, но и знать, как загрузить систему вручную. К последнему средству приходится прибегать при возникновении проблем, делающих невозможной автоматическую загрузку. Это может быть вызвано, например, повреждением файловой системы или ошибками в конфигурации сетевой платы.

Этапы загрузки

Типичная процедура начальной загрузки системы Linux состоит из шести этапов:

- загрузка и инициализация ядра;
- обнаружение и конфигурирование устройств;
- создание процессов ядра;
- действия оператора (только при ручной загрузке)
- выполнение сценариев запуска;
- работа в многопользовательском режиме.

Почти все этапы не требуют контроля со стороны администратора. Можно управлять загрузкой системы, редактируя сценарии запуска.

Инициализация ядра

Подробно о ядре рассказывается в главе 28.

Ядро Linux представляет собой программу, и первым этапом начальной загрузки является запись этой программы в память для последующего выполнения. Файл ядра обычно называется `/vmlinuz` или `/boot/vmlinuz`.

В Linux загрузка ядра осуществляется в два этапа. Сначала в память компьютера с диска считывается (с помощью кода, записанного в ПЗУ) небольшая программа начальной загрузки, которая затем выполняет собственно загрузку ядра.

Ядро выполняет тесты, позволяющие определить, сколько памяти имеется в распоряжении системы. Часть внутренних структур ядра имеет фиксированный размер, поэтому ядро резервирует определенный объем физической памяти для самого себя. Эта память недоступна пользовательским процессам. Ядро выдает на консоль сообщение об общем объеме физической памяти и объеме памяти, доступной пользователю.

Конфигурирование аппаратных средств

Одна из первых задач ядра — выявление компонентов аппаратного обеспечения. Создавая ядро для конкретной системы, можно указать, какие устройства оно должно проверять. При активизации ядро пытается найти и проинициализировать все устройства, о которых ему было сообщено. Ядро выводит на консоль краткую информацию о каждом обнаруженном устройстве. Современные дистрибутивы включают ядро, которое способно работать в большинстве аппаратных конфигураций компьютеров, требуя при этом минимальной настройки или же вообще обходясь без нее.

Информация об устройствах, задаваемая при конфигурировании ядра, зачастую оказывается неполной. В подобной ситуации ядро пытается получить необходимые сведения, опрашивая устройства, подключенные к системной шине, и собирая нужную информацию у соответствующих драйверов. Драйверы отсутствующих или не отвечающих на контрольный сигнал устройств отключаются. Если позднее устройство будет подключено к системе, можно будет динамически загрузить или активизировать его драйвер. Подробно этот процесс описан в главе 28.

Процессы ядра

После завершения этапа базовой инициализации ядро создает в области памяти, выделенной для пользовательских программ, несколько “самовыполняющихся” процессов. Это происходит “в обход” стандартного системного вызова `fork` (описан в разделе 4.2).

Подробнее списки `ps` описаны в разделе 4.7.

Количество и характер таких процессов зависят от операционной системы. В Linux это демон `init` (всегда имеет идентификатор 1; процесс с идентификатором 0 отсутствует) и различные обработчики памяти и сигналов ядра, в том числе те, которые представлены в табл. 2.1. Все эти процессы имеют идентификаторы с низкими номерами, а их имена в списках `ps` заключены в квадратные скобки (например, `[kacpid]`). Иногда имена процессов могут содержать в конце символ косой черты и цифру, как, например, `[kblockd/0]`. Число указывает процессор, на котором выполняется данный процесс. Эта информация может быть полезна при настройке многопроцессорной системы.

Таблица 2.1. Некоторые наиболее часто встречающиеся процессы ядра Linux

Процесс	Назначение
`kjournald`	Записывает обновления журнала ext3 на диск[a]
`kswapd`	Выполняет подкачку процессов при недостаточном объеме физической памяти
`kreclaimd`	Освобождает страницы памяти, которые давно не использовались
`ksoftirqd`	Выполняет обработку нескольких уровней мягких прерываний
`khubd`	Выполняет конфигурирование USB-устройств

[a] Для каждой смонтированной файловой системы ext3 существует один `kjournald`.

Из этих процессов только `init` является полноценным пользовательским процессом; остальные фактически представляют собой части ядра, которые были сделаны процессами из концептуальных соображений.

После создания перечисленных процессов ядро больше не принимает участия в процедуре начальной загрузки системы. К этому моменту, однако, еще не создан ни один из процессов, управляющих базовыми операциями (например, регистрацией пользователей в системе), и большинство демонов не запущено. Обо всем этом позаботится (в некоторых случаях косвенно) демон `init`.

Действия оператора (только при ручной загрузке)

📖 Подробнее об учетной записи root рассказывается в главе 3.

Если систему нужно запустить в однопользовательском режиме, оператор устанавливает в командной строке специальный флаг (слово “single”), а ядро передает эту информацию демону `init`. Последний, в свою очередь, передает управление команде `sulogin` — специальной версии команды `login`, — которая выдает приглашение ввести пароль пользователя root[1]. Если он введен правильно, запускается интерпретатор команд с правами суперпользователя. Можно не задавать пароль, а просто нажать <Ctrl+D>, после чего загрузка продолжится в многопользовательском режиме. Подробнее различные режимы загрузки описаны в главе 3.

📖 Подробнее о файловых системах и их монтировании речь пойдет в главе 5.

В однопользовательском режиме команды выполняются почти так же, как и в полностью загруженной системе. Однако в SUSE, Debian и Ubuntu обычно монтируется только корневой раздел. Чтобы можно было использовать программы, находящиеся вне каталогов `/bin`, `/sbin` или `/etc`, необходимо вручную смонтировать остальные файловые системы.

Во многих однопользовательских средах корневой каталог файловой системы монтируется в режиме только для чтения. Если `/tmp` является частью корневой файловой системы, выполнение множества команд, которые используют временные файлы (таких как `vi`), будет невозможно. Чтобы выйти из положения, придется начать однопользовательский сеанс с повторного монтирования каталога `/` в режиме чтения/записи. Нужное действие выполняет команда `# mount -o rw,remount /`.

[1] Подробнее эти вопросы рассмотрены на страницах интерактивного руководства (man-страницах) по командам `inittab` и `sulogin`. К сожалению, даже в современных версиях Red Hat и Fedora по умолчанию ввод пароля для загрузки в однопользовательском режиме не требуется.

В системах Red Hat и Fedora загрузка в однопользовательском режиме выполняется несколько более активно, нежели обычно. До того момента, когда на экран будет выведено приглашение интерпретатора команд, эти дистрибутивы попытаются смонтировать все локальные файловые системы. Хотя на первый взгляд такой подход кажется удобным, но при использовании недостаточно продуманной файловой системы он может порождать проблемы.

Команда `fsck`, которая проверяет и восстанавливает поврежденные файловые системы, обычно выполняется в ходе автоматической загрузки. Если система запускается в однопользовательском режиме, команду `fsck` придется вводить вручную. Подробнее эта команда описана в разделе 8.5.

Когда интерпретатор команд однопользовательского режима завершит свою работу, система продолжит загрузку в многопользовательском режиме.

Выполнение сценариев запуска системы

В тот момент, когда система сможет выполнять сценарии запуска, ее уже можно назвать Linux. Это еще не полностью загруженная система, но "загадочных" этапов процесса загрузки больше не осталось. Файлы сценариев представляют собой обычные командные файлы, которые выбираются и запускаются демоном `init` по сложному, но, в общем-то, понятному алгоритму.

Точное местонахождение, содержимое и организация сценариев запуска заслуживают отдельного изучения. Подробнее об этом рассказывается в разделе 2.5.

Работа в многопользовательском режиме

🕮 Подробнее процесс регистрации в системе описан в разделе 27.9.

После выполнения сценариев запуска система полностью готова к работе, за одним исключением: никто не может в нее войти. Для того чтобы с конкретного терминала (включая системную консоль) можно было зарегистрироваться в системе, должен быть запущен процесс `getty`, ожидающий поступления запросов от этого терминала. Демон `init` порождает все необходимые процессы `getty`, заканчивая этап начальной загрузки. Если система сконфигурирована для работы в графическом режиме, демон `init` также порождает соответствующие регистрационные процессы, такие как `xdm` или `gdm`.

Не забывайте о том, что демон `init` продолжает играть важную роль даже по завершении начальной загрузки. У него есть один однопользовательский и несколько многопользовательских "уровней выполнения", определяющих, какие ресурсы системы будут доступны пользователю. Уровни выполнения рассматриваются в разделе 2.5.

2.2. Загрузка системы на персональном компьютере

До этого момента описывалась общая схема начальной загрузки. Теперь некоторые наиболее важные (и сложные) ее этапы необходимо рассмотреть подробнее, проанализировав дополнительные особенности функционирования персональных компьютеров.

Загрузка системы на персональном компьютере — это многоступенчатый процесс. Когда включается компьютер, начинает выполняться код, записанный в ПЗУ. Точное его местонахождение и структура зависят от типа оборудования. В компьютерах, созданных специально для UNIX или другой коммерческой операционной системы, код

"прошивается" разработчиком, который заранее задает алгоритм подключения устройств, базовой инициализации сети и распознавания локальных файловых систем. Это очень удобно для системного администратора. Ему достаточно ввести лишь имя нового файла ядра, а код ПЗУ автоматически обнаружит и прочитает этот файл.

На персональных компьютерах код начальной загрузки представлен в виде базовой подсистемы ввода-вывода — BIOS (Basic Input/Output System), которая чрезвычайно упрощена в сравнении с фирменным кодом UNIX-станций. В действительности в BIOS есть несколько уровней кода: для самого компьютера, для видеоплаты, для SCSI-адаптера, если таковой имеется, и, иногда, для других периферийных устройств вроде сетевых плат.

Встроенному коду BIOS известно об устройствах, расположенных на материнской плате, в частности контроллере IDE (и жестких дисках), плате сетевого адаптера, контроллере клавиатуры, последовательных и параллельных портах. SCSI-адаптеры знают только об устройствах, подключенных непосредственно к ним. К счастью, за последние несколько лет сложные взаимодействия, необходимые для обеспечения совместной работы этих устройств, были стандартизованы и теперь почти не требуют вмешательства оператора.

В современных компьютерах код BIOS интеллектуальнее, нежели раньше. Они позволяют на этапе начальной загрузки входить в режим конфигурирования, удерживая нажатой одну или две клавиши. В большинстве случаев названия этих клавиш отображаются на экране, чтобы их не нужно было искать в документации.

В режиме конфигурирования можно выбрать, с какого устройства требуется загружаться, хотя выбор не так уж велик. Обычно последовательность загрузки задается в виде правила, например: "сначала — дисковод для гибких дисков, затем — дисковод CD-ROM, в последнюю очередь — жесткий диск". К сожалению, в некоторых BIOS загрузка ограничена возможностью загрузки с первого IDE-дисковода CD-ROM или первого жесткого диска IDE. Могут также распознаваться SCSI-адаптеры.

Когда компьютер определил, с какого устройства следует загружаться, считываются первые 512 байтов с диска. Этот сегмент диска называется главной загрузочной записью (ГЗЗ). В ней хранится программа, которая сообщает компьютеру о том, в каком разделе диска расположена программа вторичной загрузки (загрузчик операционной системы). Дополнительная информация о разделах дисков на персональных компьютерах и главной загрузочной записи приведена в главе 7.

Стандартная программа, находящаяся в ГЗЗ, дает компьютеру указание извлечь загрузчик ОС из первого раздела диска. В Linux поддерживаются более сложные программы, которые могут работать с несколькими операционными системами и ядрами.

Найдя раздел, с которого будет выполняться загрузка системы, программа ГЗЗ пытается запустить загрузочную программу, связанную с этим разделом. В случае успеха этой программе передаются полномочия по дальнейшей загрузке ядра.

2.3. Утилиты начальной загрузки: LILO и GRUB

В реальной жизни всегда есть выбор. Сегодня мы имеем возможность выбирать между двумя популярными загрузчиками: LILO и GRUB. LILO — это традиционный загрузчик Linux, очень стабильный и снабженный хорошей документацией. Но в настоящее время он быстро вытесняется загрузчиком GRUB, который является используемым по умолчанию в системах Red Hat, SUSE и Fedora. Фактически дистрибутивы Red Hat и Fedora вообще не содержат LILO. С другой стороны Debian продолжает использовать LILO.

GRUB: универсальный загрузчик

Этот загрузчик может быть использован в любой Linux-системе и особенно популярен среди пользователей, которые устанавливают несколько операционных систем на одном компьютере. Им также пользуются разработчики ядра. Загрузчик GRUB полезен, когда конфигурация системы часто меняется. В отличие от загрузчика LILO, который при каждом изменении его конфигурации должен быть повторно записан в ГЗЗ или в загрузочный сектор, GRUB читает свой конфигурационный файл в ходе начальной загрузки, поэтому дополнительный этап не нужен.

GRUB инсталлируется на загрузочный диск с помощью команды `grub-install`, которая в качестве аргумента принимает имя загрузочного устройства. К сожалению, у загрузчика GRUB используются свои правила именования физических дисков, отличающиеся от стандартных соглашений, принятых в Linux (хотя GRUB может применять также и стандартные имена Linux). Имя устройства в загрузчике GRUB выглядит примерно так:

```
(hd0,0)
```

Первое числовое значение определяет номер физического диска (нумерация ведется с нуля), а второе — номер раздела (правила нумерации аналогичны). В данном примере запись `(hd0,0)` соответствует устройству `/dev/hda1`. Следовательно, если нужно инсталлировать загрузчик GRUB на основной жесткий диск, воспользуйтесь такой командой:

```
# grub-install '(hd0,0)'
```

Кавычки необходимы, чтобы интерпретатор команд не пытался интерпретировать скобки по-своему.

По умолчанию GRUB читает параметры загрузки из файла `/boot/grub/grub.conf`. Вот его образец:

```
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.6.9-5)
      root (hd0,0)
      kernel /boot/vmlinuz-2.6.9-5 ro root=/dev/hda1
```

В этом примере конфигурируется единственная операционная система, которая будет загружена автоматически (`default=0`), если в течение 10 секунд с клавиатуры не поступят никакие данные (`timeout=10`).

Корневая файловая система для конфигурации Red Hat Linux находится в разделе `(hd0,0)`. GRUB загрузит ядро из файла `/boot/vmlinuz-2.6.9-5`, а затем выведет начальный образ экрана, хранящийся в файле `/boot/grub/splash.xpm.gz`.

GRUB поддерживает мощный интерфейс командной строки и ряд функций, которые позволяют редактировать записи файла конфигурации в ходе загрузки. Чтобы перейти в режим командной строки, в окне загрузки GRUB необходимо ввести команду `c`. Из командной строки можно загружать операционные системы, не отраженные в файле `grub.conf`, выводить на экран информацию о системе и выполнять простейшую проверку файловой системы. Загрузчик предоставляет также ряд функций, подобных функциям интерпретатора команд, в том числе — функции завершения не полностью введенных команд и перемещения курсора. Любые действия, которые могут быть выполнены с помощью файла `grub.conf`, могут быть выполнены и посредством командной строки загрузчика GRUB.

Нажатие клавиши <Tab> позволяет вывести на экран краткий список доступных команд. Некоторые из наиболее полезных команд перечислены в табл. 2.2.

Таблица 2.2. Параметры командной строки GRUB

Команда	Назначение
`reboot`	Мягкая перезагрузка системы
`find`	Поиск файла во всех смонтированных логических разделах
`root`	Указание корневого устройства (логического раздела)
`kernel`	Загрузка ядра с корневого устройства
`help`	Вывод интерактивной справки по команде
`boot`	Загрузка системы с указанного образа ядра

Для получения подробной информации о загрузчике GRUB и его параметрах командной строки обратитесь к официальному руководству:

`www.gnu.org/software/grub/manual/`

LILO: традиционный загрузчик Linux

Конфигурирование и инсталляция загрузчика осуществляется с помощью команды `lilo`, которая извлекает параметры конфигурации из файла `/etc/lilo.conf`. Чтобы изменить настройки загрузчика, нужно отредактировать этот файл и повторно выполнить команду `lilo`. Эту процедуру необходимо проделывать всякий раз при изменении процесса загрузки — в частности, когда добавляется новый загрузочный раздел или создается новое ядро.

Загрузчик LILO может быть установлен в главную загрузочную запись диска или в загрузочную запись корневого раздела Linux.

Ниже приведено содержимое файла `lilo.conf` для системы, в которой имеются рабочее и резервное ядра:

```
boot=/dev/hda            # помещаем загрузчик в ГЗЗ
root=/dev/hda1           # задаем корневой раздел
install=/boot/boot.b
map=/boot/map
delay=20 # двухсекундная задержка, дающая пользователю возможность вмешаться
image=/vmlinuz           # загружаемое ядро
  label=linux            # метка ядра
  read-only
image=/vmlinuz-backup    # резервное ядро
  label=backup
  read-only
```

Каждому возможному сценарию загрузки присваивается метка. Введя метку на этапе загрузки, можно сообщить программе LILO о том, какой из сценариев следует выбрать. Сценарий, который указан в файле `lilo.conf` первым, выбирается по умолчанию.

В стандартном сценарии (метка `linux`) загружается файл `/vmlinuz`. Флаг `read-only` указывает на то, что ядро монтирует свою корневую файловую систему в режиме только для чтения. Этот флаг должен всегда присутствовать. Сценарии запуска системы позаботятся о том, чтобы повторно смонтировать раздел в режиме чтения/записи, когда воз-

никнет такая необходимость. Система сконфигурирована таким образом, чтобы в случае сбоя загружалось резервное ядро (файл `/vmlinuz-backup`). Наличие подобной возможности никогда не помешает.

Если ввести команду `lilo` без аргументов, она создаст и инсталлирует программу загрузки, сообщив о том, какие ядра доступны. Название основного ядра будет помечено звездочкой. Ошибки, присутствующие в файле `lilo.conf`, не будут обнаружены до тех пор, пока процедура инсталляции загрузчика не достигнет середины. В этом случае система окажется в переходном состоянии. *Не перезагружайте* ее, пока очередной запуск команды `lilo` не даст положительных результатов.

Чтобы не попасть в подобную ситуацию, воспользуйтесь командой `lilo -t`, которая протестирует файл, не выполняя инсталляцию. Если ошибок не выявлено, переходите к процедуре инсталляции. Честно говоря, непонятно, почему команда `lilo` не делает такую проверку автоматически.

В случае показанного выше конфигурационного файла результаты работы команды будут такими:

```
# lilo
Added linux*
Added backup
```

При загрузке системы программа LILO выдаст следующее приглашение:

```
LILO:
```

После двухсекундной паузы (параметр `delay`, равный 1, соответствует десятой доле секунды, а в рассматриваемом файле `lilo.conf` он равен 20) будет загружено ядро `/vmlinuz`, а в качестве корневого будет смонтирован первый раздел первого IDE-диска. Список сценариев загрузки можно просмотреть, нажав клавишу <Tab>:

```
LILO: <Tab>
linux  backup
LILO:
```

Чтобы загрузить нужное ядро, введите его метку в строке приглашения.

Параметры ядра

Программы загрузчиков LILO и GRUB позволяют передавать ядру параметры командной строки. Как правило, эти параметры изменяют значения параметров ядра, вынуждают его опросить конкретные устройства, указывают путь поиска демона `init` или назначают конкретное корневое устройство. Несколько примеров этих параметров приведены в табл. 2.3.

Таблица 2.3 Примеры параметров ядра времени выполнения

Параметр	Назначение
`init=/sbin/init`	Сообщает ядру путь к демону `init`
`init=/bin/bash`	Заставляет ядро запускать только интерпретатор `bash`; используется при восстановлении системы в случае сбоев
`root=/dev/foo`	Сообщает ядру о том, что корневым является устройство `/dev/foo`
`single`	Задает режим однопользовательской загрузки

Мультисистемная загрузка на персональном компьютере

Поскольку на одном персональном компьютере может быть установлено несколько операционных систем, привычной является ситуация, когда компьютер загружается в мультисистемном режиме. Для этого загрузчик должен быть правильно сконфигурирован, чтобы распознать имеющиеся на локальных дисках операционные системы.

В каждом разделе диска может храниться собственный вторичный загрузчик, однако главная загрузочная запись только одна. Поэтому необходимо решить, какой из загрузчиков будет "главным". Как правило, выбор диктуется особенностями имеющихся операционных систем. Если одной из них является Linux, то лучше всего в качестве главного загрузчика выбрать LILO или GRUB. Заметим, что в данной конкретной ситуации загрузчик GRUB проявляет себя лучше, чем LILO.

Мультисистемное конфигурирование GRUB

Конфигурирование загрузчика GRUB для выполнения мультисистемой загрузки во многом аналогично конфигурированию загрузки только одной операционной системы. Прежде чем вносить изменения в файл `/boot/grub/grub.conf`, необходимо установить необходимые операционные системы.

Записанные в файле `grub.conf` параметры конфигурации для загрузки ОС Windows, отличаются от параметров для загрузки UNIX или Linux:

```
title Windows XP
       rootnoverify (hd0,0)
       chainloader +1
```

Параметр `chainloader` загружает утилиту начальной загрузки из указанного места (в данном случае из сектора 1 первого раздела первого IDE-диска). Параметр `rootnoverify` предотвращает попытки загрузчика GRUB выполнить монтирование указанного раздела. Он избавляет загрузчик GRUB от необходимости проверять непонятные для него разделы, такие как разделы NTFS или размещенные вне области, в которой GRUB может выполнять чтение.

Ниже приведен полный текст файла `grub.conf` для случая, когда Windows XP загружается из первого раздела, Linux — из второго, а Fedora — из третьего:

```
default=0
timeout=5
splashimage=(hd0,2)/boot/grub/splash.xpm.gz
hiddenmenu
title Windows XP
       rootnoverify (hd0,0)
       chainloader +1
title Red Hat
       root (hd0,1)
       kernel /boot/vmlinuz
title Fedora
       root (hd0,2)
       kernel /boot/vmlinuz
```

Мультисистемное конфигурирование LILO

Если LILO является главным загрузчиком (например, на компьютере установлены Linux и Windows XP), начните со стандартного процесса конфигурирования LILO, описанного выше. Затем по мере необходимости можно добавлять записи для других операционных систем в файл `/etc/lilo.conf`.

Вот как будет выглядеть запись, предназначенная для загрузки Windows из первого раздела первого IDE-диска:

```
other = /dev/hda1
label = windows
table = /dev/hda
```

Ниже приведен полный текст файла `lilo.conf` для случая, когда Windows загружается из первого раздела, Linux — из второго, а FreeBSD — из третьего:

```
boot = /dev/hda          # помещаем загрузчик в ГЗЗ первого IDE-диска
delay = 20 # двухсекундная задержка, дающая пользователю возможность вмешаться
default = linux          # по умолчанию загружается Linux из второго раздела
image = /boot/vmlinuz-2.6.9
       root = /dev/hda2
       label = linux
       read-only
other = /dev/hda1        # загрузка из первого раздела
       label = windows
       table = /dev/hda
other = /dev/hda3        # загрузка из третьего раздела
       label = freebsd
       table = /dev/hda
```

После модификации файла `lilo.conf` нужно повторно ввести команду `lilo`. Не забудьте предварительно выполнить ее в тестовом режиме с флагом `-t`. Более подробно разбивка диска на разделы рассматривается в разделе 7.4.

Поставщики (или добровольцы) часто распространяют пакеты исправления ошибок дистрибутивов Linux. Это же относится и к ядру. Устранение брешей в системе безопасности, исправление ошибок и добавление функций происходит регулярно. Однако, в отличие от других программных пакетов, пакеты исправления ошибок ядра не обновляются, а устанавливаются наряду с существующим ядром. Этот подход позволяет администраторам отменять обновление, если пакет устранения ошибок ядра приводит к разрушению системы. Поэтому постепенно меню начальной загрузки LILO и GRUB заполняются различными версиями ядра. Как правило, можно выполнять загрузку той операционной системы, которая выбрана по умолчанию, но помните о существовании этого простого способа разрешения проблемы в случае невозможности загрузки после установки пакета исправлений.

2.4. Загрузка в однопользовательском режиме

Однопользовательский режим прекрасно подходит для изменения конфигурации системы или выполнения задач по ее обслуживанию, не беспокоясь при этом о возможности помешать другим пользователям (или о том, что они могут помешать). Кроме того, этот режим может быть единственно доступным при работе в поврежденной системе.

📖 Подробнее уровни выполнения описаны в разделе 2.5 далее в главе.

Чаще всего переход в однопользовательский режим выполняют путем передачи аргументов программе начальной загрузки. Однако обычно переход в однопользовательский режим с другого уровня выполнения можно выполнить с помощью команды `telinit 1`. При этом перезагрузку можно не выполнять, если только не требуется устранять какую-либо проблему начальной загрузки.

В качестве меры предосторожности на случай нестабильности системы корневой каталог файловой системы монтируется в режиме только для чтения. Это, конечно же, неудобно, если требуется исправить конфигурационный файл или другой файл, находящийся в корневой файловой системе, или если нужно выполнить команду, модифицирующую файлы. Чтобы выйти из положения, начните однопользовательский сеанс с повторного монтирования каталога / в режиме чтения/записи. Нужное действие выполняет команда

```
# mount -o remount -w /
```

Конкретные процедуры перехода в однопользовательский режим при использовании загрузчиков GRUB и LILO различаются между собой.

Однопользовательский режим при использовании GRUB

Чтобы выполнить загрузку в однопользовательском режиме при использовании загрузчика GRUB, не нужно применять опции командной строки. Авторы этого загрузчика пришли к выводу, что параметры начальной загрузки должны легко поддаваться изменению, и что клавиша ‘a’ — вполне подходящее средство для решения этой задачи. Когда откроется экран начальной загрузки GRUB, выделите нужное ядро и нажмите клавишу ‘a’, чтобы дополнить опции начальной загрузки. Чтобы обеспечить загрузку в однопользовательском режиме, добавьте флаг `single` в конец существующих опций ядра. Пример типичной конфигурации мог бы выглядеть следующим образом:

```
grub append> ro root=LABEL=/ rhgb quiet single
```

Однопользовательский режим при использовании LILO

Различные дистрибутивы предлагают различные средства открытия командного приглашения LILO. Если вы отдали предпочтение LILO, а не GRUB, в системах Red Hat, Fedora или SUSE, опцию меню командной строки можно выбрать в причудливом графическом интерфейсе пользователя. Пользователи Debian и Ubuntu должны нажать и удерживать нажатой клавишу shift сразу после того, как BIOS завершит проверку памяти и другие тесты самопроверки.

В строке приглашения LILO введите метку ядра (она задана в файле `lilo.conf`), которое требуется загрузить, а затем — опцию `-s` или `single`. Например, стандартное ядро дистрибутива Debian имеет метку “linux”, поэтому, чтобы загрузиться в однопользовательском режиме, необходимо ввести следующее:

```
LILO: linux single
```

2.5. Работа со сценариями запуска системы

После выхода из однопользовательского режима (или — при автоматической загрузке — по завершении работы интерпретатора команд, запущенного с правами пользова-

теля root) демон init выполняет сценарии запуска системы. Они являются сценариями интерпретатора sh (на самом деле bash), а их местонахождение и содержимое зависят от системы.

Ниже приведен перечень задач, которые часто выполняются этими сценариями:

- задание имени компьютера;
- установка часового пояса;
- проверка дисков с помощью команды fsck (только в автоматическом режиме);
- монтирование дисков систем;
- удаление старых файлов из каталога /tmp;
- конфигурирование сетевых интерфейсов;
- запуск демонов и сетевых служб.

Сценарии запуска обычно довольно многословны и выводят подробную информацию о выполняемых ими задачах. Это может оказать существенную помощь при отладке сценариев или поиске причин зависания системы в процессе загрузки.

В старых системах нередко приходилось модифицировать сценарии, чтобы настроить их для конкретной среды. Однако разбиение программного обеспечения на узкоспециализированные пакеты и частая публикация их обновлений в Internet привели к применению более надежного подхода. Современные системы включают в себя множество небольших сценариев запуска системы, устанавливаемых различными программами, и эти сценарии считывают информацию о своей локальной конфигурации из отдельных файлов. Как правило, файлы локальной конфигурации имеют форму мини-сценариев интерпретатора sh, которые устанавливают значения переменных интерпретатора. Затем эти переменные уточняются сценариями.

Демон init и уровни выполнения

Традиционно демон init определяет 7 уровней выполнения, на каждом из которых должен выполняться конкретный набор системных служб.

- Уровень 0 означает, что система полностью прекратила работу.
- Уровень 1 или S означает однопользовательский режим.
- Уровни 2–5 предназначены для многопользовательского режима.
- Уровень 6 определяет этап перезагрузки системы.

Уровни 0 и 6 характерны тем, что система не может в них оставаться, а переходит на них в процессе завершения работы или перезагрузки. В многопользовательском режиме система чаще всего находится на уровне 2 или 3. Уровень 5 используется регистрационными процессами X Window, например программой xdm. Уровень 4 используется редко, а уровни 1 и S различны для каждой системы.

Однопользовательскому режиму традиционно соответствовал уровень 1. На нем запрещены все многопользовательские сеансы и процессы удаленной регистрации, а в системе выполняется минимальный набор программ. Но поскольку в этом режиме доступ к системе осуществляется с правами пользователя root, администраторам необходимо, чтобы при загрузке система выдавала приглашение на ввод пароля. Для этой цели был введен уровень S: здесь создается отдельный процесс, отображающий требуемое приглашение. В Linux уровень S носит переходный характер и завершается сразу после ввода пароля.

Создается впечатление, что уровней выполнения больше, чем нужно. Обычно это объясняется тем, что в телефонном коммутаторе 7 уровней, поэтому считалось, что в UNIX-системе должно быть как минимум столько же. В действительности в Linux поддерживается до десяти уровней, хотя уровни 7–9 не определены.

В файле `/etc/inittab` содержатся параметры, определяющие, что должен делать демон `init` на каждом из уровней. Формат файла зависит от системы, но основная идея состоит в том, что в нем задаются команды, которые должны быть выполнены (или продолжить выполнение), когда система переходит на конкретный уровень.

В процессе загрузки демон `init` последовательно продвигается от уровня 0 к уровню по умолчанию, заданному в файле `/etc/inittab`. Чтобы осуществить переход между соседними уровнями, демон выполняет команды из этого файла. Аналогичные действия, только в обратном порядке, происходят при останове системы.

К сожалению, структура файла `inittab` несколько устарела. Чтобы сделать этот файл более эффективным, Linux-системы реализуют дополнительный, абстрактный уровень. Он представлен в виде сценария (обычно `/etc/init.d/rc`), который запускается из файла `inittab` и осуществляет смену уровней. Этот сценарий, в свою очередь, выполняет другие сценарии из каталога, зависящего от целевого уровня. Они-то и переводят систему в новое состояние.

Большинство современных дистрибутивов Linux по умолчанию выполняет загрузку на 5 уровень, что может быть не подходящим для серверов, которым не требуется запускать X Window. Изменение используемого по умолчанию уровня выполнения не представляет сложности. Следующий фрагмент файла `inittab` компьютера, работающего под управлением системы SUSE, определяет 5 уровень выполнения в качестве используемого по умолчанию:

```
id:5:initdefault:
```

Системным администраторам обычно не приходится работать непосредственно с файлом `/etc/inittab`, так как существующие сценарии подходят для большинства случаев. Далее в главе мы не будем упоминать об этом файле и взаимодействии демона `init` со сценариями запуска системы. Просто, когда мы говорим о том, что демон `init` выполняет такой-то сценарий, нужно понимать: связь со сценарием может быть косвенной.

Основные копии сценариев запуска хранятся в каталоге `/etc/init.d`. Каждый сценарий отвечает за запуск одного демона или определенной подсистемы. Сценариям можно передавать аргументы `start` и `stop`, которые означают, что соответствующая служба должна быть запущена либо остановлена. Большинство сценариев понимает также аргумент `restart`, который эквивалентен связке `stop+start`. Обладая правами системного администратора, можно вручную запускать или останавливать нужные службы, вызывая соответствующий сценарий из каталога `init.d` и передавая ему требуемый аргумент.

Ниже показан несложный сценарий, позволяющий запускать, останавливать и перезапускать демон `sshd`.

```
#! /bin/sh
test -f /usr/bin/sshd || exit 0
case "$1" in
  start)
  echo -n "Starting sshd: sshd"
  /usr/sbin/sshd
  echo "."
  ;;
```

```
  stop)
  echo -n "Stopping sshd: sshd"
  kill `cat /var/run/sshd.pid`
  echo "."
  ;;
default=0
timeout=10
splashimage=(hd0,0)/boot/grub/splash.xpm.gz
title Red Hat Linux (2.6.9-5)
      root (hd0,0)
      kernel /boot/vmlinuz-2.6.9-5 ro root=/dev/hda1
```

Сценарии каталога `/etc/init.d` способны запускать и останавливать отдельные службы, но, чтобы перейти на требуемый уровень, главный управляющий сценарий, выполняемый демоном `init`, должен получить дополнительную информацию о том, какие сценарии и с какими аргументами нужно запустить. Управляющий сценарий не просматривает непосредственно каталог `init.d`, а обращается к каталогу `rc`*уровень*`.d`, где *уровень* — это номер требуемого уровня выполнения, на который осуществляется переход (`rc0.d`, `rc1.d` и т.д.).

В каталогах `rc`*уровень*`.d` обычно содержатся символические ссылки на сценарии каталога `init.d`. Имена ссылок начинаются с префикса `S` или `K`, за которым следует номер и имя службы, управляемой сценарием (например, `S34named`). Если демон `init` переходит на более высокий уровень, он выполняет все сценарии с префиксом `S` ("start" — запуск) в порядке возрастания номеров, причем каждому сценарию передается аргумент `start`. Когда осуществляется переход на более низкий уровень, запускаются сценарии с префиксом `K` ("kill" — уничтожить) в порядке убывания номеров, и всем им передается аргумент `stop`.

Эта схема позволяет администраторам очень точно управлять порядком запуска служб. Например, запуск службы SSH до запуска сетевых интерфейсов лишен смысла. Хотя в системе Fedora и сеть, и служба выполняются на 2 уровне, сценарию `network` присваивается порядковый номер 10, а сценарию `sshd` — 55. Поэтому можно не сомневаться, что сценарий `network` будет запущен раньше. При добавлении новых служб необходимо учитывать подобные взаимосвязи.

Чтобы сообщить системе, когда следует запускать тот или иной демон, нужно создать символическую ссылку в соответствующем каталоге. Например, следующие команды информируют систему о том, что демон `CUPS` должен быть запущен на уровне 2 и остановлен при завершении работы системы:

```
# ln -s /etc/init.d/cups /etc/rc2.d/S80cups
# ln -s /etc/init.d/cups /etc/rc0.d/K80cups
```

Первая ссылка свидетельствует о том, что сценарий `/etc/init.d/cups` должен быть запущен одним из последних при переходе на уровень 2 и ему должен быть передан аргумент `start`. Вторая ссылка сообщает, что в процессе завершения работы системы сценарий `/etc/init.d/cups` должен быть запущен относительно поздно, причем с аргументом `stop`. В некоторых системах процессы останова и перезагрузки трактуются по-разному, поэтому необходимо также создать символическую ссылку в каталоге `/etc/rc6.d`, чтобы обеспечить корректный останов демона при перезагрузке системы.

Сценарии запуска Red Hat и Fedora

Исторически так сложилось, что в сценариях Red Hat и Fedora непросто разобраться, так как в них могут присутствовать комментарии вида

```
# Несколько громоздкое, но необходимое действие
```

На каждом уровне выполнения демон `init` вызывает сценарий `/etc/rc.d/rc`, передавая ему номер уровня в качестве аргумента. Этот сценарий может выполняться как в обычном режиме, так и в режиме подтверждения, когда перед выполнением каждого сценария выдается запрос.

Управлять службами можно с помощью команды `chkconfig`. Эта команда добавляет или удаляет сценарии запуска из системы, управляет уровнями выполнения, на которых они работают, и выводит уровни, для которых сконфигурирован сценарий. Для получения информации о применении этого простого и удобного средства можно использовать команду `man chkconfig`.

В Red Hat есть также сценарий `rc.local`, напоминающий одноименный сценарий в BSD-системах. В процессе начальной загрузки он выполняется последним. В предыдущих версиях Red Hat файл `rc.local` перезаписывался пакетом `initscripts`. Однако теперь ситуация изменилась, и в этот файл можно безбоязненно помещать собственные параметры запуска.

Ниже приведен пример сеанса запуска системы Red Hat:

```
[kernel information]
INIT: version 2.85 booting
Setting default font (latarcyrhev-sun16):                    [  OK  ]
      Welcome to Red Hat Linux
  Press 'I' to enter interactive startup.
Starting udev:                                               [  OK  ]
Initializing hardware... storage network audio done
Configuring kernel parameters:                               [  OK  ]
Setting clock  (localtime): Tue Mar 29 20:50:41 MST 2005:    [  OK  ]
...
```

Когда появится сообщение “Welcome to Red Hat Enterprise Linux”, можно нажать клавишу <i>, чтобы продолжить загрузку в режиме подтверждения. Однако подтверждение о нажатии клавиши не отображается. Система продолжит монтировать локальные файловые системы, активизировать разделы подкачки, загружать таблицы клавиш и вести поиск модулей ядра. Только после перехода на уровень 3 демон `init` начнет выдавать запросы:

```
       Welcome to Red Hat Enterprise Linux WS
         Press 'I' to enter interactive startup.
Starting udev:                                               [  OK  ]
Initializing hardware... storage network audio done
Configuring kernel parameters:                               [  OK  ]
setting clock  (localtime): tue mar 29 20:50:41 mst 2005:    [  OK  ]
Setting hostname rhel4:                                      [  OK  ]
Checking root filesystem
/dev/hda1: clean, 73355/191616 files, 214536/383032 blocks   [  OK  ]
Remounting root filesystem in read-write mode:               [  OK  ]
Setting up Logical Volume Management:                        [  OK  ]
Checking filesystems
Mounting local filesystems:                                  [  OK  ]
```

```
Enabling local filesystem quotas:                          [  OK  ]
Enabling swap space:                                       [  OK  ]
INIT: Entering runlevel: 3
Entering interactive startup
Start service kudzu (Y)es/(N)o/(C)ontinue? [Y]
```

Режимы интерактивной и однопользовательской загрузки начинаются в одной и той же точке. Если в процессе загрузки возникли серьезные проблемы и этой точки достичь невозможно, воспользуйтесь загрузочной дискетой или компакт-диском.

Можно передать загрузчику LILO параметр `init=/bin/sh`, чтобы заставить его вызвать интерпретатор команд однопользовательского режима еще до того, как будет запущен демон `init`[2]. В этом случае все действия по запуску системы придется осуществлять вручную, включая выполнение команды `fsck` и монтирование локальных файловых систем.

Повлиять на процесс начальной загрузки в Red Hat можно путем модификации конфигурационных файлов, находящихся в каталоге `/etc/sysconfig`. Назначение элементов этого каталога описано в табл. 2.4

Таблица 2.4. Файлы и подкаталоги каталога `/etc/sysconfig` в Red Hat

Файл/подкаталог	Назначение или содержимое
`clock`	Задает тип системных часов (почти всегда UTC)[a]
`console`	Загадочный каталог, который всегда пуст
`httpd`	Определяет модель обработки Apache, которую необходимо использовать
`hwconf`	Содержит всю информацию о системных аппаратных средствах; используется службой Kudzu
`i18n`	Содержит региональные установки системы (форматы представления даты/времени, языки и т.д.)
`init`	Определяет способ отображения сообщений, поступающих от сценариев запуска системы
`keyboard`	Задает тип клавиатуры (используйте идентификатор “us” для стандартной 101-клавишной клавиатуры)
`mouse`	Задает тип мыши; используется системой X и программой `gpm`
`network`	Задает глобальные сетевые опции (имя компьютера, шлюз, переадресация и т.д.)
`network-scripts`	Каталог, содержащий вспомогательные сценарии и сетевые конфигурационные файлы
`sendmail`	Задает параметры программы `sendmail`

[a] При использовании мультисистемной загрузки совершенно непонятно, какой часовой пояс следует устанавливать для системных часов.

Некоторые элементы списка заслуживают дополнительных пояснений.

[2] Однажды в нашей системе был поврежден файл, содержащий таблицу клавиш, и поскольку система Red Hat загружала этот файл даже в однопользовательском режиме, переход в этот режим не позволял решить проблему. Передача программе LILO параметра `init=/bin/sh` оказалась единственным безопасным способом загрузить систему и исправить ошибку. Этот прием эффективен и в других ситуациях.

- Файл `hwconf` содержит всю информацию об имеющемся оборудовании. Он просматривается службой Kudzu, которая проверяет, было ли добавлено или удалено какое-нибудь устройство, и запрашивает у пользователя дополнительные инструкции. В системах промышленного уровня эту службу можно отключить, поскольку она существенно задерживает процесс начальной загрузки. Каждый раз, когда обнаруживается изменение аппаратной конфигурации, возникает задержка длительностью 30 секунд.
- Каталог `network-scripts` содержит вспомогательные файлы, связанные с конфигурацией сети. Единственное, что может потребоваться в нем изменить, — это файлы с именами `ifcfg-`*интерфейс*. Например, файл `network-scripts/ifcfg-eth0` хранит параметры конфигурации интерфейса `eth0`, в частности его IP-адрес и сетевые опции. Подробнее конфигурирование сетевых интерфейсов рассматривается в разделе 12.7.
- Файл `sendmail` содержит две переменные: `DAEMON` и `QUEUE`. Если переменная `DAEMON` равна `yes`, система запустит программу `sendmail` в режиме демона (`-bd`) в процессе начальной загрузки. Переменная `QUEUE` информирует программу `sendmail` о том, каков интервал обработки очереди почтовых сообщений. Стандартная установка — 1 час.

Сценарии запуска SUSE

Хотя сценарии запуска системы SUSE подобны сценариям RHEL и Fedora, это область, в которой SUSE превосходит другие дистрибутивы Linux. Эти сценарии четко организованы, надежны и хорошо документированы. Стоит сказать спасибо разработчикам, которые отвечают за эту часть операционной системы.

Как и в системах Red Hat и Fedora, на каждом уровне выполнения демон `init` вызывает сценарий `/etc/rc.d/rc`, передавая ему номер уровня в качестве аргумента. Сценарии данного пакета хранятся в каталоге `/etc/init.d`, а их файлы конфигурации — в каталоге `/etc/sysconfig`. Прекрасное краткое описание процесса запуска системы SUSE можно найти в файле `/etc/init.d/README`.

Хотя файлы конфигурации запуска как системы SUSE, так и RHEL/Fedora собраны в каталоге `/etc/sysconfig`, конкретные файлы внутри этого каталога в значительной мере различны. (В частности, в целом файлы SUSE снабжены достаточно подробными комментариями.) Необходимые параметры вызываются посредством установки переменных среды интерпретатора, к которым затем обращаются сценарии, хранящиеся в каталоге `/etc/init.d`. Некоторые подсистемы требуют более подробного конфигурирования. Такие системы, нуждающиеся в нескольких файлов конфигурации, имеют приватные подкаталоги вроде `sysconfig/network`.

Файл `windowmanager` — типичный пример файла, хранящегося в каталоге `sysconfig`:

```
## Path:                Desktop/Window manager
## Description:
## Type:                string(kde,fvwm,gnome,windowmaker)
## Default:             kde
## Config:              profiles,kde,susewm
#
# Here you can set the default window manager (kde, fvwm, ...)
# changes here require at least a re-login
```

```
DEFAULT_WM="kde"

## Type:                yesno
## Default:             yes
# install the SUSE extension for new users
# (theme and additional functions)
#
INSTALL_DESKTOP_EXTENSIONS="yes"
```

Каждой переменной предшествует информация о конфигурации, считываемая утилитой YaST[3], и подробное описание назначения переменной. Например, в файле `windowmanager` переменная `DEFAULT_WM` определяет диспетчер окна рабочего стола, используемый системой X.

Особенно удачно решение SUSE о помещении файлов конфигурации сети в подкаталог `/etc/sysconfig/network`. Этот каталог содержит как файлы глобальной конфигурации (с параметрами настройки которые относятся ко всем сетевым интерфейсам), так и файлы конфигурации конкретных сетевых интерфейсов. Например, файл `network/routes` содержит глобальную информацию маршрутизации. В типичной системе SUSE содержимое этого файла может выглядеть следующим образом:

```
# Destination       Dummy/Gateway     Netmask       Device
default             192.168.10.254    0.0.0.0       eth0
```

Маршруты, которые должны присутствовать только во время работы конкретного интерфейса, можно указывать в файле `ifroute-`*имя*. Например, в интерфейсе `eth1` этот файл имел бы имя `ifroute-eth1`, а его содержимым могло бы быть:

```
# Destination       Dummy/Gateway     Netmask       Device
10.10.0.0/24        10.10.0.254
```

При желании можно указать маску сети и устройство, но сценарии запуска будут определять правильные значения.

Пакет SUSE содержит также команду `chkconfig`, которая позволяет управлять сценариями запуска системы. Она коренным образом отличается о версии, предоставляемой системой Red Hat, но также является эффективным средством управления сценариями.

Независимо от того, используется утилита YaST либо команда `chkconfig` или сценарии запуска системы обрабатываются вручную, желательно просмотреть файл `/etc/sysconfig` и проанализировать его содержимое.

Типичный сеанс запуска системы SUSE может выглядеть следующим образом:

```
[kernel information]
INIT: version 2.85 booting
System Boot Control: Running /etc/init.d/boot
Mounting /proc filesystem                                          done
Mounting sysfs on /sys                                             done
Mounting /dev/pts                                                  done
Boot logging started on /dev/tty1(/dev/console) at Tue Mar 29 14:04:12 2005
Mounting shared memory FS on /dev/sh                               done
Activating swap-devices in /etc/fstab...
Adding 1052248k swap on /dev/hda2. Priority:42 extents:1           done
Checking root file system...
...
```

[3] YaST — специализированная утилита графической конфигурации SUSE, которая управляет многими аспектами этой системы. Более подробно утилита YaST рассматривается в главе 11.

Сценарии запуска Debian и Ubuntu

Если SUSE является примером системы с четко продуманным планом управления сценариями запуска, то Debian — ее антипод. Сценарии Debian ненадежны, недокументированны и невероятно противоречивы. Печально, но отсутствие стандартных правил организации сценариев запуска привело в данном случае к хаосу.

На каждом уровне выполнения демон `init` вызывает сценарий `/etc/init.d/rc`, передавая ему номер уровня в качестве аргумента. Все сценарии должны самостоятельно искать свою конфигурационную информацию, которая хранится в файлах каталога `/etc`, `/etc/default` или другого подкаталога каталога `/etc` либо в самих сценариях.

Имя компьютера хранится в файле `/etc/hostname`, который читается сценарием `/etc/init.d/hostname.sh`. Параметры сетевых интерфейсов и стандартных шлюзов находятся в файле `/etc/network/interfaces`, который читается командой `ifup`, вызываемой из сценария `/etc/init.d/networking`. Некоторые сетевые опции могут быть также заданы в файле `/etc/network/options`.

Системы Debian и Ubuntu предоставляют своего рода скрытую программу управления сценариями запуска — `update-rc.d`. Хотя посвященная ей страница руководства предостерегает против ее интерактивного использования, мы сочли ее полезной, хотя и менее удобной, заменой команды `chkconfig`. Например, для запуска `sshd` на уровнях выполнения 2, 3, 4 и 5 и для останова на уровнях 0, 1 и 6 можно использовать следующую команду:

```
$ sudo /usr/sbin/update-rc.d sshd start 0123 stop 456
```

2.6. Перезагрузка и останов системы

Буферы файловых систем Linux хранятся в памяти и лишь изредка записываются на диск. Это ускоряет выполнение операций дискового ввода-вывода, но повышает риск потери данных в случае внезапного сбоя.

Традиционные UNIX- и Linux-системы были очень требовательны в отношении процедуры выключения. Современные системы более терпимы (особенно если речь идет о такой высоконадежной файловой системе, как ext3fs), но все же лучше корректно завершать работу, если это возможно. Неправильное выключение компьютера может привести к появлению трудно обнаруживаемых, неочевидных ошибок, а иногда и к полному краху системы.

Перезагрузка системы на персональном компьютере — средство решения почти всех проблем. Но при работе в Linux советуем сначала подумать и только потом перезагружаться. Проблемы, возникающие в Linux, как правило, скрытые и сложные, поэтому перезагрузка дает ожидаемый результат гораздо реже, чем в других системах. Кроме того, процесс перезагрузки Linux занимает много времени, что создает неудобства для пользователей.

Перезагружаться необходимо в том случае, когда подключается новое устройство или работающее устройство зависает так, что его невозможно проинициализировать. Если модифицируется конфигурационный файл, который опрашивается только при начальной загрузке, то изменения вступят в силу лишь после перезагрузки. И, наконец, если в системе невозможно зарегистрироваться, иного выхода, кроме как перезагрузиться, просто не существует.

Если модифицируется один из сценариев запуска системы, то нужно перезагрузиться хотя бы для того, чтобы проверить, успешно ли функционирует система после изменений. Если какая-либо проблема не проявится в течение нескольких ближайших недель, впоследствии вы вряд ли вспомните подробности последних изменений.

В отличие от начальной загрузки, которая осуществляется единственным способом, остановить и перезагрузить систему можно по-разному:

- выключить питание;
- ввести команду `shutdown`;
- использовать команды `halt` и `reboot`;
- изменить уровень выполнения демона `init` с помощью команды `telinit`;
- выполнить команду `poweroff`, чтобы попросить систему выключить питание.

Выключение питания

Даже в системах настольных компьютеров выключение питания — не лучший способ останова системы. Это может привести к потере данных и повреждению файловых систем.

В некоторых компьютерах имеется кнопка программного останова, при нажатии которой выполняется ряд команд, корректно завершающих работу системы. Если вы не уверены, поддерживает ли компьютер такую возможность, не пытайтесь это выяснить, нажав кнопку выключения питания в процессе работы системы! Будет гораздо меньше проблем, если остановить систему вручную.

Конечно, предусмотрительность хороша в разумных пределах. В случае наводнения или пожара лучше отключить питание, если на корректный останов системы просто нет времени. Когда-то в машинных залах существовала аварийная кнопка, которая позволяла выключить все оборудование одновременно.

Команда `shutdown`: корректный способ останова системы

Команда `shutdown` — самый безопасный и корректный способ остановить или перезагрузить систему либо вернуться в однопользовательский режим.

Можно дать команде указание делать паузу перед остановом системы. Во время ожидания команда посылает зарегистрированным пользователям через постепенно укорачивающиеся промежутки времени сообщения, предупреждая о приближающемся событии. По умолчанию в сообщениях говорится о том, что система заканчивает работу, и указывается время, оставшееся до момента останова. При желании администратор может добавить собственное короткое сообщение, в котором поясняется, почему система останавливается, и сколько примерно времени придется подождать, прежде чем снова можно будет войти в систему. После выполнения команды `shutdown` пользователи будут лишены возможности входа в систему, но они будут видеть сообщение, предусмотренное администратором.

С помощью команды `shutdown` можно указать, что должна сделать система после выполнения команды: остановиться (`-h`) или перезагрузиться (`-r`). Можно также задать, должна ли после перезагрузки выполняться принудительная проверка дисков с помощью команды `fsck` (`-F`) или нет (`-f`). По умолчанию Linux автоматически пропускает эту проверку, если файловые системы были корректно демонтированы.

Следующая команда напоминает пользователям о запланированной процедуре сервисного обслуживания и отключает систему в 9:30 утра:

```
# shutdown -h 09:30 "Going down for scheduled maintenance.
      Expected downtime is 1 hour"
```

Можно также задать относительное время отключения. Например, приведенная ниже команда запустит процесс выключения через 15 минут:

```
# shutdown -h +15 "Going down for emergency disk repair."
```

Команда `halt`: более простой способ останова

Команда `halt` выполняет все основные операции, необходимые для останова системы.

Обычно она вызывается командой `shutdown -h`, но может применяться и сама по себе. Команда регистрирует в журнальном файле факт останова, уничтожает несущественные процессы, выполняет системный вызов `sync`, дожидается завершения операций записи на диск, а затем прекращает работу ядра.

При наличии опции `-n` системный вызов `sync` подавляется. Команда `halt -n` используется после восстановления корневого раздела командой `fsck`, чтобы ядро не могло затереть исправления старыми версиями раздела, хранящимися в кэше.

Команда `reboot`: быстрый перезапуск

Команда `reboot` почти идентична команде `halt`. Разница лишь в том, что система перезагружается, а не останавливается. Режим перезагрузки вызывается также командой `shutdown -r`. Команда `reboot` тоже поддерживает флаг `-n`.

Команда `telinit`: изменение уровня выполнения демона `init`

С помощью команды `telinit` можно дать демону `init` указание перейти на конкретный уровень выполнения. Например, команда

```
# telinit 1
```

переводит систему в однопользовательский режим.

В отличие от команды `shutdown`, команда `telinit` не поддерживает такие возможности, как предупреждающее сообщение и пауза на подготовку, поэтому ее стараются избегать. Обычно она используется при проверке изменений, внесенных в файл `inittab`.

Команда `poweroff`: запрос на выключение питания

Команда `poweroff` идентична команде `halt`, за исключением того, что после останова операционной системы посылается запрос системе управления питанием (если таковая имеется) с указанием выключить питание компьютера. Это позволяет дистанционно выключать машины.

Эквивалентная команда `poweron` отсутствует. Причины должны быть всем очевидны.

2.7. Упражнения

2.1. Почему так важно выполнить команду `lilo -t` перед инсталляцией загрузчика LILO? Как загрузить ядро, имя которого не `vmlinuz`?

2.2. Почему Linux-системы нельзя выключать нажатием кнопки питания на корпусе компьютера? Какие есть альтернативы?

2.3. Используйте опции командной строки загрузчика GRUB для загрузки ядра, информация о котором отсутствует в файле `grub.conf`.

☆ 2.4. Объясните концепцию уровней выполнения. Перечислите уровни выполнения, существующие в Linux, и кратко опишите каждый из них. Какова связь между уровнями `1` и `S`?

☆ 2.5. Напишите сценарий, предназначенный для запуска некоего сетевого демона `/usr/local/sbin/foo`. Покажите, как организовать автоматический запуск сценария на этапе начальной загрузки системы.

☆ 2.6. Найдите и инсталлируйте программу `mactime` Дэна Фармера и Витса Венема (является частью пакета TCT). Запустите ее для создания начальной базы данных с метками времени системных файлов. Перезагрузите компьютер. Запустите программу `mactime` повторно и определите, какие файлы были модифицированы в процессе загрузки. Какие файлы были просмотрены, но не модифицированы? (Необходим доступ с правами суперпользователя.)

☆☆ 2.7. Если система находится на уровне выполнения 4 и вводится команда `telinit 1`, какие действия предпримет демон `init`? Каким будет конечный результат?

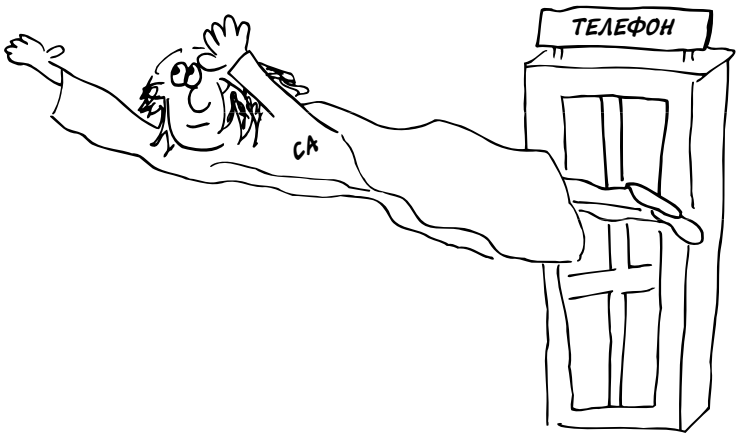
☆☆ 2.8. Нарисуйте схему, показывающую очередность запуска демонов в вашей Linux-системе.

☆☆ 2.9. Опишите последовательность действий, необходимых для инсталляции на одном компьютере операционных систем Linux и Windows XP/2000. Используйте программу GRUB и загрузчик Windows.

Book_LinuxAdmHand.indb 84 17.04.2007 13:54:37

Глава 3

Сила привилегий



Каждый файл и процесс в Linux принадлежит определенному пользователю. Не имея соответствующих привилегий, другие пользователи не могут получить доступ к чужим объектам. Такая схема позволяет защитить объекты каждого пользователя и предотвратить несанкционированный доступ, случайный или злонамеренный.

Чаще всего системными файлами и процессами владеет фиктивный пользователь root, известный как *суперпользователь*. Все его ресурсы надежно защищены от несанкционированного вмешательства. Чтобы иметь возможность выполнять административные задачи, необходимо тем или иным способом зарегистрироваться в системе как суперпользователь, о чем и пойдет речь в этой главе.

Суперпользователь обладает "магической" властью. В частности, он имеет право выступать в качестве владельца любого файла или процесса и выполнять специальные операции, недоступные рядовым пользователям. Привилегированный пользователь — очень ответственная должность, а если ее занимает беспечный человек или злоумышленник, то она становится еще и очень опасной.

В этой главе рассматриваются основы привилегированного доступа к системе. О том, как предотвратить несанкционированный доступ на уровне суперпользователя, рассказывается в главе 20. В главе 30 описываются административные аспекты этого процесса.

3.1. Владение файлами и процессами

Каждый файл в Linux принадлежит владельцу и группе. Владелец файла имеет особую привилегию, недоступную другим пользователям системы: ему разрешено менять права доступа к файлу. В частности, владелец может задать права доступа так, что никто,

кроме него, не сможет обращаться к файлу[1]. Мы еще вернемся к теме прав доступа в главе 5.

Дополнительную информацию о группах можно найти в главе 6.

Владельцем файла всегда является один человек, тогда как в группу владельцев могут входить несколько пользователей. По традиции информация о группах хранилась в файле `/etc/group`, но теперь ее чаще хранят на сетевом сервере NIS или LDAP. Подробнее эти вопросы рассматриваются в главе 17.

Владелец файла должен определить, какие операции могут выполнять над файлом члены группы. Такая схема допускает совместное использование файлов членами одной группы. Например, мы применяем группу для управления доступом к файлам исходного кода Web-сайта `www.admin.com`

Узнать идентификаторы владельцев файла можно с помощью команды `ls -l` *`имя_файла`*. Например:

```
$ ls -l /staff/scott/todo
-rw-------   1  scott   staff    1258  Jun 4 18:15   /staff/scott/todo
```

Как несложно заметить, файлом владеет пользователь `scott`, а группа, которой он принадлежит, называется `staff`.

Операционная система Linux отслеживает не имена владельцев и групп, а их идентификаторы. В самом общем случае идентификаторы пользователей (сокращенно UID — User ID) и соответствующие им имена хранятся в файле `/etc/passwd`, а идентификаторы и названия групп находятся в файле `/etc/group`. Текстовые эквиваленты идентификаторов определяются исключительно для удобства пользователей. Чтобы команда вроде `ls` могла вывести информацию о принадлежности файла в удобочитаемом виде, она должна просмотреть базу данных идентификаторов и найти в ней нужные имена.

Владелец может посылать процессу сигналы (описываются в разделе 4.3), а также понижать его приоритет. С процессорами связаны, по меньшей мере, семь идентификаторов: реальный, эффективный и сохраненный идентификатор пользователя, реальный, эффективный и сохраненный идентификатор группы, и, в системе Linux — "идентификатор пользователя файловой системы", который используется только для определения прав доступа к файлам. Вообще говоря, реальные номера применяются для учета использования системных ресурсов, а эффективные — для указания прав доступа. В большинстве случаев реальные и эффективные идентификаторы совпадают.

Сохраненные идентификаторы не оказывают никакого непосредственного влияния. Они позволяют программам "приберегать" неактивные идентификаторы для последующего использования, тем самым способствуя расчетливому применению расширенных полномочий. Вообще говоря, идентификатор пользователя файловой системы — это особенность реализации файловой системы NFS, и обычно он совпадает с эффективным идентификатором пользователя.

О битах прав доступа рассказывается в разделе 5.5.

Процесс не может явно изменить ни один из своих идентификаторов, но есть особая ситуация, когда новые эффективные идентификаторы устанавливаются косвенно. Дело в том, что существуют два специальных бита, устанавливаемых в маске прав доступа к файлу: "setuid" (Set User ID — бит смены идентификатора пользователя) и "setgid" (Set

[1] Более того, можно сделать так, что файл станет недоступным даже для владельца.

Group ID — бит смены идентификатора группы). Если запускается исполняемый файл, у которого установлен один из этих битов, то эффективными идентификаторами создаваемого процесса становятся идентификаторы владельца файла, а не идентификаторы пользователя, запустившего программу. Смена полномочий действительна только на время работы программы.

Бит "setuid" позволяет рядовым пользователям запускать программы, выполняющие жестко регламентированные административные действия. Например, команда `passwd`, с помощью которой пользователь меняет свой пароль, обращается к файлу `/etc/shadow` (или `/etc/passwd`), принадлежащему суперпользователю, вследствие чего у нее установлен бит "setuid". Она модифицирует файл строго определенным образом и завершается. Конечно, даже столь простое действие может стать причиной злоупотреблений, поэтому, прежде чем внести запрашиваемые изменения, команда `passwd` проверяет, знает ли пользователь свой текущий пароль.

3.2. Суперпользователь

Определяющей характеристикой учетной записи суперпользователя является значение UID, равное нулю. Linux не запрещает менять имя этой учетной записи или создавать другую запись с нулевым идентификатором, но такие действия ни к чему хорошему не приведут. Их следствием будет возникновение новых брешей в системе защиты, а также растерянность и гнев других администраторов, которым придется разбираться с особенностями конфигурирования такой системы.

Традиционная система UNIX позволяет суперпользователю (т.е. всякому процессу, эффективный идентификатор пользователя которого равен нулю) выполнять над файлом или процессом любую допустимую операцию[2]. Кроме того, некоторые системные вызовы (обращения к ядру) разрешено осуществлять только суперпользователю. Вот примеры операций, доступных лишь суперпользователю:

- изменение корневого каталога процесса с помощью команды `chroot`;
- создание файлов устройств;
- установка системных часов;
- увеличение лимитов использования ресурсов и повышение приоритетов процессов[3];
- задание сетевого имени компьютера;
- конфигурирование сетевых интерфейсов;
- открытие привилегированных сетевых портов (номера которых меньше 1024);
- останов системы.

Процессы суперпользователя обладают способностью изменять свои идентификаторы. Один из таких процессов — программа `login` и ее графические эквиваленты, которые отображают на экране приглашение ввести пароль при входе в систему. Если введенные пароль и имя пользователя правильны, программа заменяет свои идентификаторы

[2] Следует подчеркнуть важность слова "допустимую". Некоторые операции (например, запуск файла, для которого не установлен бит выполнения) запрещены даже суперпользователю.

[3] Начиная с версии ядра 2.6.12, новый лимит использования ресурсов позволяет пользователям, которые не являются суерпользователем, повышать приоритеты провесов, если это разрешено системным администратором.

фикаторы соответствующими идентификаторами указанного пользователя и запускает интерпретатор команд. После того как процесс суперпользователя, сменив владельца, станет обычным пользовательским процессом, восстановить свое предыдущее привилегированное состояние он уже не сможет.

Теоретически системы Linux позволяют распределять привилегии учетной записи суперпользователя по "возможностям" в соответствии со стандартом POSIX. По различным причинам, в том числе из-за проблем, присущих текущей реализации, эта функция оказалась не столь полезной или ценной для системных администраторов, как ожидалось вначале. Подробнее эти возможности рассмотрены в разделе 20.6.

3.3. Пароль суперпользователя

📖 Подробнее о взломе паролей читайте в разделе 20.10.

Пароль пользователя root должен состоять как минимум из восьми символов, так как даже семисимвольные пароли взламываются достаточно легко. В системах, где применяются пароли DES, задавать более длинный пароль не имеет смысла, потому что обрабатываются только первые восемь символов. В начале главы 6 рассказывается о том, как разрешить использование паролей MD5, которые могут иметь большую длину.

Пароль суперпользователя следует выбирать так, чтобы его нельзя было угадать или определить методом последовательного перебора. Теоретически наиболее безопасный пароль состоит из случайного набора букв, знаков препинания и цифр. Такой пароль, однако, тяжело запомнить и, как правило, неудобно вводить, поэтому, если системный администратор записывает пароль на бумажке или вводит его слишком медленно, об оптимальном уровне безопасности системы говорить не приходится.

До недавнего времени достаточно надежным и удобным для запоминания считался пароль, состоящий из двух произвольно выбранных слов, разделенных знаком препинания. К сожалению, теперь такие пароли взламываются очень быстро, и мы не рекомендуем их использовать.

Сегодня наиболее широкое распространение получил подход, заключающийся в выборе пароля по принципу "шокирующего абсурда". Этот принцип был определен Грейди Уордом (Grady Ward) в ранней версии списка часто задаваемых вопросов, посвященного выбору идентификационной фразы для PGP.

> *Принцип "шокирующего абсурда" заключается в составлении короткой фразы (или предложения), которая лишена смысла и в то же время вызывает шок у пользователя в данной социальной среде. То есть она должна содержать совершенно неприличные или расистские высказывания либо состоять из абсолютно не стыкующихся выражений. Применять такой подход не предосудительно, поскольку подразумевается, что пароль никогда не станет известен людям, чьи чувства могут быть им оскорблены.*
>
> *Маловероятно, чтобы такой пароль был повторен кем-то еще, ведь он уникален по своей сути. При этом он легко запоминается, потому что имеет яркую эмоциональную окраску. Сдержанный пример "шокирующего абсурда" выглядит так: "Моллюски отгрызли мои гарцующие гениталии". Любой читатель без труда придумает гораздо более шокирующие фразы.*

Преобразовать подобную фразу в пароль можно, записав только первые буквы каждого слова или применив какое-нибудь другое легко запоминающееся преобразование.

Безопасность пароля значительно возрастет, если включить в него цифры, знаки препинания и прописные буквы.

Пароль суперпользователя следует менять:

- минимум каждые три месяца;
- всякий раз, когда кто-либо, знающий пароль, увольняется из организации;
- когда, по вашему мнению, безопасности системы что-то угрожает;
- если вы не планируете провести вечер так бурно, что на следующее утро рискуете не вспомнить пароль.

3.4. Как стать суперпользователем

Поскольку суперпользователь является таким же членом системы, как и остальные пользователи, можно войти в систему непосредственно под именем root. Однако оказывается, что это довольно неудачное решение. Во-первых, не будет сделано никаких записей о том, какие операции выполнял суперпользователь. Согласитесь, не слишкомто приятно обнаружить, что вчера ночью в 3:00 вы что-то поменяли, но никак не можете вспомнить, что именно. Еще хуже, если такой доступ был несанкционированным и необходимо выяснить, какой ущерб системе нанес нарушитель. Во-вторых, сценарий регистрации суперпользователя не предполагает сбора дополнительной идентифицирующей информации. Когда под именем root в систему могут входить несколько пользователей, не существует способа определить, кто именно из них и когда это делал.

Вследствие упомянутых причин в большинстве систем регистрация под именем root может быть запрещена на терминалах и по сети, т.е. везде, кроме системной консоли[4]. Мы рекомендуем следовать этой схеме. В разделе 20.9 рассказывается о том, какие системные файлы потребуется для этого отредактировать.

Команда `su`: замена идентификатора пользователя

Лучше получать доступ к учетной записи root с помощью команды `su`. Будучи вызванной без аргументов, она выдает приглашение на ввод пароля суперпользователя, а затем запускает интерпретатор команд с правами пользователя root. Интерпретатор будет выполняться в привилегированном режиме, пока не завершит работу (при нажатии клавиш <Ctrl+D> или по команде `exit`). Команда `su` не фиксирует действия, производимые в среде интерпретатора, но добавляет запись в журнальный файл с указанием, кто и когда вошел в систему под именем суперпользователя.

Команда `su` способна также подставлять вместо имени root имена других пользователей. Иногда единственный способ помочь пользователю в решении проблемы — войти с помощью команды `su` в его среду и разобраться в происходящем, так сказать, “на месте”.

Зная чей-либо пароль, можно непосредственно зарегистрироваться в системе под его именем, введя команду `su` *`имя_пользователя`*. В ответ будет выдан запрос на ввод пароля. Или другой вариант: сначала стать суперпользователем, воспользовавшись командой `su`, а затем с помощью этой же команды перейти в другую учетную запись. В таком случае вводить пароль не потребуется.

[4] Система Ubuntu Linux пошла еще дальше. По умолчанию система не содержит никакого действующего пароля для учетной записи root, и в ней необходимо использовать команду `sudo`, подробно описанную в последующих абзацах этого раздела.

Рекомендуем взять за правило при вводе команды указывать полное имя, например /bin/su, а не просто su. Это послужит определенной защитой от тех программ с именем su, которые преднамеренно были прописаны в переменной среды PATH злоумышленником, намеревавшимся собрать хороший "урожай" паролей[5].

Утилита sudo: ограниченный вариант команды su

Поскольку полномочия суперпользователя разделить (по крайней мере, произвольным образом) нельзя, трудно предоставить кому-то право выполнять конкретную административную операцию (например, создавать резервные копии), не предоставив возможность свободно работать в системе. Если же учетная запись root доступна целой группе администраторов, то вы и понятия не будете иметь о том, кто ею пользуется и что он при этом делает.

Наиболее часто используемое решение заключается в использовании утилиты sudo. По умолчанию она входит во все рассматриваемые нами дистрибутивы, а ее исходный код можно получить на Web-сайте www.courtesan.com.

Утилита sudo в качестве аргумента принимает командную строку, которая подлежит выполнению от имени пользователя root (или другого уполномоченного пользователя). Утилита обращается к файлу /etc/sudoers, где содержится список пользователей, имеющих разрешение на ее выполнение, и перечень команд, которые они могут вводить на конкретном компьютере. Если запрашиваемая команда разрешена, утилита sudo приглашает пользователя ввести *его собственный* пароль, и выполняет команду от имени суперпользователя.

Далее утилита sudo позволяет, не вводя пароль, выполнять другие команды, но только до тех пор, пока не наступит пятиминутный период бездействия (его продолжительность можно менять). Такая мера служит защитой от тех привилегированных пользователей, которые оставляют свои терминалы без присмотра.

Утилита sudo ведет журнал, где регистрируются выполненные команды, компьютеры, на которых они выполнялись, и вызвавшие их пользователи, а также каталоги, из которых запускались команды, и время их вызова. Эта информация может направляться в систему Syslog или сохраняться в любом журнальном файле по усмотрению пользователя. Мы рекомендуем направлять ее на защищенный центральный компьютер.

Строка журнального файла, содержащая данные о пользователе randy, который выполнил команду sudo /bin/cat etc/sudoers, может выглядеть следующим образом:

```
Dec 7 10:57:19 tigger sudo: randy: TTY=ttyp0 ; PWD=/tigger/users/randy;
       USER=root ; COMMAND=/bin/cat /etc/sudoers
```

Существует единая версия файла /etc/sudoers, которая используется на всех компьютерах. Сам файл выглядит примерно так:

```
# Псевдонимы для факультетов физики и вычислительной техники
Host_Alias CS = tigger, anchor, piper, moet, sigi
Host_Alias PHYSICS = eprince, pprince, icarus
```

[5] По аналогичной причине мы настоятельно рекомендуем не включать запись "." (текущий каталог) в переменную path интерпретатора команд. Несмотря на очевидное удобство, подобная конфигурация приводит к тому, что можно непреднамеренно запустить "специальную" версию какой-нибудь системной команды, оставленной злоумышленником в качестве приманки. Пользователю root следует быть бдительным вдвойне.

```
# Набор команд
Cmnd_Alias      DUMP = /sbin/dump, /sbin/restore
Cmnd_Alias      PRINTING = /usr/sbin/lpc, /usr/bin/lprm
Cmnd_Alias      SHELLS = /bin/sh, /bin/tcsh, /bin/csh, /bin/bash, /bin/ash,

# Права доступа
mark, ed        PHYSICS = ALL
herb            CS = /usr/sbin/tcpdump : PHYSICS = (operator) DUMP
lynda           ALL = (ALL) ALL, !SHELLS
%wheel          ALL, !PHYSICS = NOPASSWD: PRINTING
```

Первые пять строк (не считая комментариев) определяют набор компьютеров и команд, на которые имеются ссылки в спецификациях прав доступа. Списки элементов можно включать в спецификации в полном виде, но работать с псевдонимами проще, к тому же они делают файл `/etc/sudoers` понятнее, и в него легче вносить изменения. Разрешается также создавать псевдонимы для различных групп пользователей.

В каждую спецификацию прав доступа включается информация о:

- пользователях, к которым относится запись;
- компьютерах, на которых пользователям разрешено выполнять соответствующие действия;
- командах, которые могут выполняться указанными пользователями;
- пользователях, от имени которых могут выполняться команды.

Первая строка спецификаций применяется к пользователям mark и ed, которые регистрируются в системе на компьютерах группы `PHYSICS` (eprince, pprince и icarus). Встроенный псевдоним `ALL` разрешает им вводить любые команды. Поскольку дополнительный список пользователей в скобках не указан, утилита `sudo` будет выполнять команды только от имени суперпользователя.

Пользователь herb может запускать утилиту `tcpdump` на компьютерах группы `CS`, а также выполнять команды резервного копирования на компьютерах группы `PHYSICS`. Обратите внимание, однако, что вторая группа команд получит привилегии не пользователя root, а пользователя operator. Реальная команда, которую пришлось бы вести пользователю herb, выглядит примерно так:

```
$ sudo -u operator /sbin/dump 0u /dev/hda2
```

Пользователь lynda имеет право выполнять команды от имени любого пользователя на любом компьютере, но она не может запускать большинство распространенных интерпретаторов команд. Означает ли это, что она не может получить доступ к интерпретатору? Конечно, нет:

```
$ cp -p /bin/bash /tmp/bash
$ sudo /tmp/bash
```

Вообще говоря, попытка разрешить “все команды, кроме...” обречена на провал, по крайней мере, с технической точки зрения. Тем не менее, имеет смысл использовать подобную форму спецификации, так как это послужит хотя бы напоминанием о том, что вызывать интерпретатор команд в режиме суперпользователя не рекомендуется, и предотвратит случайные попытки вызова.

В последней строке пользователям группы wheel разрешается выполнять команды печати lpc и lprm от имени суперпользователя на всех компьютерах, за исключением eprince, pprince и icarus. Более того, от пользователей не требуется вводить пароль.

Обратите внимание на то, что команды в файле /etc/sudoers приводятся с указанием полного имени, чтобы пользователи не могли выполнять свои собственные программы и сценарии от имени суперпользователя. Разрешается также указывать допустимые аргументы команд (в приведенных выше примерах это не показано). В целом возможности файла sudoers очень велики, а рассмотренная конфигурация иллюстрирует лишь основные из них.

Для модификации файла /etc/sudoers предназначена специальная команда visudo, которая проверяет, не редактируется ли файл кем-то посторонним, затем открывает его в редакторе, а перед инсталляцией файла выполняет синтаксический контроль. Последний этап особенно важен, поскольку ошибка в файле /etc/sudoers может не позволить повторно вызвать утилиту sudo для исправления файла.

Использование утилиты sudo имеет следующие преимущества:

- благодаря регистрации команд значительно повышается степень административного контроля над системой;
- операторы могут выполнять специальные задачи, не имея неограниченных привилегий;
- настоящий пароль суперпользователя могут знать всего один-два человека;
- вызывать утилиту sudo быстрее, чем выполнять команду su или входить в систему под именем root;
- пользователя можно лишить привилегий, не меняя пароль суперпользователя;
- ведется список всех пользователей с правами пользователя root;
- меньше вероятность того, что интерпретатор команд, запущенный суперпользователем, будет оставлен без присмотра;
- управлять доступом ко всей сети можно с помощью одного файла.

📖 Дополнительная информация об утилите John the Ripper приведена в разделе 20.10.

Утилита sudo имеет и недостатки. Самый большой из них заключается в том, что любая брешь в системе защиты того или иного привилегированного пользователя эквивалентна нарушению безопасности самой учетной записи root. Противостоять этому нелегко. Можно лишь предупредить тех, кто имеют право выполнять утилиту sudo, о необходимости максимально защищать свои учетные записи. Можно также регулярно запускать утилиту John the Ripper, проверяя стойкость паролей привилегированных пользователей.

Другой недостаток — возможность обмануть утилиту sudo с помощью таких уловок, как временный выход в интерпретатор команд из разрешенной программы либо выполнение команд sudo sh или sudo su, если они допустимы.

3.5. Другие псевдопользователи

Только пользователь root имеет для ядра Linux особый статус. Есть, однако, еще несколько псевдопользовательских учетных записей, которые применяются для системных целей. Пароли этих псевдопользователей в файле /etc/shadow обычно заменяют звездочкой, чтобы нельзя было войти в систему под служебным именем.

Унаследованный владелец системных команд: bin

В некоторых старых UNIX-системах пользователь bin является владельцем большинства системных команд, а также каталогов, в которых они хранятся. Назначение отдельного пользователя для этих целей часто считается избыточным (и даже небезопасным), поэтому в современных системах (включая Linux) соответствующую роль в основном берет на себя пользователь root. С другой стороны, учетная запись bin считается стандартной, поэтому от нее нельзя полностью избавиться.

Владелец непривилегированных системных программ: daemon

Файлы и процессы, которые являются частью операционной системы, но не должны принадлежать пользователю root, иногда передаются во владение пользователю daemon. Считается, что это поможет избежать риска, связанного с использованием учетной записи root. По аналогичным причинам существует и группа daemon. Как и учетная запись bin, запись daemon почти не задействуется в большинстве дистрибутивов Linux.

Общий пользователь NFS: nobody

📖 Дополнительную информацию об учетной записи nobody можно найти в конце раздела 16.1.

Сетевая файловая система — NFS (Network File System) — использует учетную запись nobody для представления суперпользователей в других системах, где есть общедоступные файлы. Чтобы лишить суперпользователей их исключительных прав, излишних в данном контексте, NFS должна на время сеанса удаленного доступа заменить нулевой идентификатор чем-то другим. Этой цели как раз и служит учетная запись nobody.

Пользователю nobody не нужны специальные права доступа, и он не должен владеть никакими файлами. Если бы ему принадлежали какие-то файлы, то над ними получили бы контроль суперпользователи, регистрирующиеся в удаленных системах.

Традиционно пользователю nobody соответствовал идентификатор -1 или -2, и это соглашение по-прежнему соблюдается в ядре Linux, где идентификатор этого пользователя равен 65534 (16-битный беззнаковый эквивалент -2). В других дистрибутивах просто выбирается маленькое значение идентификатора (например, Red Hat и Fedora используют значение, равное 99), что при использовании 32-битных идентификаторов пользователей более оправдано. Единственный недостаток этого подхода в том, что, судя по всему, команда `exportfs` не обращает внимания на файл `passwd`, поэтому с помощью опции `anonuid` ей нужно явно указывать о необходимости использования другого идентификатора для пользователя nobody.

3.6. Упражнения

☆ 3.1. Воспользуйтесь командой `find` с флагом `-perm`, чтобы найти в своей системе пять файлов с установленным битом “setuid”. Объясните, почему каждая из команд не сможет правильно функционировать без этого бита.

☆ 3.2. Составьте три фразы по принципу “шокирующего абсурда”, а затем пропустите их через утилиту `md5sum` и сравните результаты. Почему безопаснее использовать пароли в формате MD5?

☆ 3.3. Опишите последовательность команд, позволяющих модифицировать чужой пароль, и покажите, как "замести следы". Предполагается, что доступ к учетной записи root имеется только через утилиту `sudo` (разрешены все команды, кроме интерпретаторов и `su`).

☆ 3.4. Создайте две записи в файле конфигурации `sudoers`.

а) Первая запись разрешает пользователям matt, adam и drew конфигурировать принтер, очищать его очередь и перезапускать демоны печати на компьютере printserver.

б) Вторая запись позволяет пользователям drew, smithgr и jimlane удалять задания и перезагружать компьютеры в студенческой лаборатории.

☆ 3.5. Инсталлируйте утилиту `sudo`, которая будет направлять вам по электронной почте сообщения о попытках несанкционированного доступа. Используйте ее для проверки записей, созданных в предыдущем упражнении, и убедитесь в том, что утилита правильно регистрирует сообщения в Syslog. (Необходим доступ с правами суперпользователя; возможно, понадобится также модифицировать файл `/etc/syslog.conf`.)

Глава 4

Управление процессами

Процесс — это абстракция, используемая в Linux для описания выполняющейся программы. Процесс представляет собой системный объект, посредством которого можно контролировать обращения программы к памяти, центральному процессору и ресурсам ввода-вывода.

Концепция, согласно которой как можно больше работы должно выполняться в контексте процессов, а не в ядре, является частью философии UNIX и Linux. Системные и пользовательские процессы подчиняются одним и тем же правилам, благодаря чему управление ими осуществляется с помощью единого набора команд.

4.1. Атрибуты процесса

Процесс состоит из адресного пространства и набора структур данных, содержащихся внутри ядра. Адресное пространство представляет собой совокупность страниц памяти[1], которые были выделены ядром для выполнения процесса. В него загружается код процесса и используемые им библиотеки функций, а также переменные, содержимое стеков и различная вспомогательная информация, необходимая ядру во время работы процесса. Поскольку Linux представляет собой систему, в которой поддерживается концепция виртуальной памяти, страницы адресного пространства процесса в конкретный момент времени могут находиться либо в физической памяти, либо в разделе подкачки, т.е. на диске.

[1] Страницы — это базовые блоки памяти, размер которых, как правило, составляет 4 Кбайт на персональных компьютерах.

В структурах данных ядра хранится всевозможная информация о каждом процессе. К наиболее важным сведениям относятся:

- таблица распределения памяти;
- текущий статус (неактивен, приостановлен, выполняется и т.п.);
- приоритет;
- информация об используемых ресурсах;
- информация о файлах и сетевых портах, открытых процессом;
- маска сигналов (запись о том, какие сигналы блокируются);
- имя владельца процесса.

Некоторые из этих атрибутов могут совместно использоваться несколькими процессами, образующими "потоковую группу", которая является Linux-аналогом многопоточного процесса в традиционной системе UNIX. Хотя эти процессы могут совместно использовать адресное пространство, члены потоковой группы обладают собственными приоритетами и статусами выполнения. На практике лишь немногие из процессов, представляющих интерес для системных администраторов, используют несколько потоков выполнения. Но даже те из них, которые используют потоки (такие как процесс `named` в BIND 9), в целом не требуют к себе особого внимания со стороны администратора на этом уровне детализации.

Многие характеристики процесса непосредственно влияют на его выполнение. В частности, имеет значение, сколько времени выделяется ему центральным процессором, к каким файлам он имеет доступ и т.д. В следующих подразделах мы рассмотрим наиболее интересные с точки зрения системного администратора характеристики процессов. Они поддерживаются во всех версиях UNIX и Linux.

Идентификатор процесса (PID)

Ядро назначает каждому процессу уникальный идентификатор (Process ID, PID). Большинство команд и системных вызовов, работающих с процессами, требует указания конкретного идентификатора, чтобы был ясен контекст операции. Идентификаторы присваиваются по порядку по мере создания процессов.

Идентификатор родительского процесса (PPID)

В Linux нет системного вызова, который создавал бы новый процесс для выполнения конкретной программы. Существующий процесс должен клонировать сам себя, чтобы породить новый процесс. Клон может заменить выполняемую программу другой.

В операции клонирования исходный процесс называют родительским, а его клон — дочерним. Помимо собственного идентификатора каждый дочерний процесс имеет атрибут PPID (Parent Process ID), который совпадает с идентификатором породившего его родительского процесса[2].

Идентификатора родительского процесса — весьма полезная информация, если приходится иметь дело с неизвестным (и, возможно, нестандартно ведущим себя) процессом. Отслеживание истоков процесса может облегчить понимание его назначения и значимости.

[2] По крайней мере, первоначально. Если родительский процесс по какой-то причине завершается раньше потомка, демон `init` (процесс с номером 1) подставляет себя на место предка (подробности описаны в разделе 4.2).

Идентификатор пользователя (UID) и эффективный идентификатор пользователя (EUID)

Дополнительная информация об идентификаторах пользователя приведена в разделе 6.1.

UID (User ID) — это идентификатор пользователя, создавшего данный процесс, точнее, копия значения UID родительского процесса. Менять атрибуты процесса могут только его создатель (владелец) и пользователь root.

EUID (Effective User ID) — это "эффективный", или текущий, пользовательский идентификатор процесса, предназначенный для того, чтобы определить, к каким ресурсам и файлам у процесса есть право доступа в данный момент. У большинства процессов значения UID и EUID одинаковы. Исключение составляют программы, у которых установлен бит смены идентификатора пользователя (setuid).

Зачем нужны два идентификатора? Просто имеет смысл разграничить понятия персонификации и прав доступа. К тому же программы с установленным битом setuid не всегда выполняются с расширенными привилегиями. Значение EUID можно устанавливать и сбрасывать, чтобы предоставлять процессу дополнительные полномочия или отбирать их.

Операционная система Linux хранит начальный идентификатор, т.е. копию значения EUID, которое имел процесс в начальной точке. Если процесс не удалит сохраненный идентификатор, его можно будет использовать в качестве реального или эффективного идентификатора. Благодаря этому "консервативно" написанная программа с установленным битом setuid может большую часть времени работать с обычными привилегиями, переходя в режим расширенных привилегий лишь в определенные моменты.

В Linux есть еще и нестандартный параметр FSUID, определяющий возможности работы с файловой системой, но вне ядра он используется редко.

Разобраться в тонкостях применения различных идентификаторов не так-то просто. Этой теме посвящен прекрасный Web-документ *Secure Programming for Linux and Unix HOWTO* написанный Дэвидом Вилером и доступный на Web-сайте `www.dwheeler.com`.

Идентификатор группы (GID) и эффективный идентификатор группы (EGID)

Дополнительная информация о группах приведена в разделе 6.1.

GID (Group ID) — это идентификатор группы, к которой принадлежит владелец процесса. Эффективный, или текущий, идентификатор группы (Effective Group ID, EGID) связан с атрибутом GID так же, как значение EUID — с UID, т.е. они будут отличаться в случае программы, у которой установлен бит смены идентификатора группы (setgid). В ядре назначение сохраненного GID аналогично назначению сохраненного атрибута UID.

В значительной степени атрибут GID процесса является рудиментарным. С точки зрения определения прав доступа в Linux процесс одновременно может относиться к нескольким группам. Список групп хранится отдельно от значений GID и EGID. При анализе прав доступа проверяется эффективный идентификатор и дополнительный список групп, но не значение GID. Если пользователь пытается получить доступ к какому-либо ресурсу, весь список проверяется на предмет того, принадлежит ли пользователь к группе, членам которой разрешается использовать данный ресурс.

Атрибут GID используется только при создании процессом новых файлов. В зависимости от установленных прав доступа в данной файловой системе новые файлы могут принимать атрибут GID создающего их процесса. Подробнее эти вопросы рассмотрены в разделе 5.5.

Приоритет и фактор уступчивости

Приоритет процесса определяет, какую долю времени центрального процессора получает программа. Ядро применяет динамический алгоритм вычисления приоритетов, учитывающий, сколько времени центрального процессора уже использовал процесс и сколько времени он ожидает своей очереди. Кроме того, учитывается заданный администратором так называемый фактор уступчивости (устанавливается с помощью команды `nice`), который определяет, в какой степени программа может "делиться" процессором с другими программами. Подробнее этот механизм рассматривается в разделе 4.6.

Для улучшения поддержки низко-латентных приложений в Linux в традиционную модель планирования задач UNIX добавлены "классы планирования". В настоящее время определения три класса планирования, и каждый процесс относится к одному из них. К сожалению, классы реального времени пока не находят широкого применения, а их поддержка из командной строки оставляет желать лучшего. Все системные процессы используют традиционный планировщик, ориентированный на применение фактора уступчивости. В этой книге мы рассмотрим только стандартный планировщик. Более подробную информацию по вопросам, связанным с планированием в реальном времени, можно найти на Web-сайте `www.realtimelinuxfoundation.org`.

Управляющий терминал

С большинством процессов, не являющихся демонами, связан управляющий терминал, который определяет базовую конфигурацию стандартных каналов ввода, вывода и ошибок. Когда пользователь запускает какую-либо программу в интерпретаторе, его терминал, как правило, становится управляющим терминалом процесса. От управляющего терминала зависит также распределение сигналов, о чем пойдет речь в разделе 4.3.

4.2. Жизненный цикл процесса

Для создания нового процесса существующий процесс, как правило, клонирует сам себя с помощью системного вызова `fork`. В результате формируется копия исходного процесса, имеющая лишь некоторые отличия. В частности, новому процессу присваивается собственный идентификатор, и учет ресурсов ведется для него независимо от предка.

Системный вызов `fork` имеет уникальное свойство: он возвращает два разных значения. В дочернем процессе это будет 0, а в родительском — идентификатор процесса-потомка. Поскольку в остальном процессы идентичны, они должны проверять результат вызова, чтобы определить, какую роль следует играть в дальнейшем.

После завершения системного вызова `fork` дочерний процесс обычно запускает новую программу с помощью одного из системных вызовов семейства `exec`[3]. Все вызовы этого семейства выполняют приблизительно одинаковые действия: они замещают сегмент кода процесса и устанавливают сегменты данных и стека в исходное состояние.

Формы вызовов `exec` различаются только способами указания аргументов командной строки и переменных среды, передаваемых новой программе.

В Linux определена альтернатива вызову `fork` — системный вызов `clone`, создающий набор процессов, которые совместно используют память или пространства ввода-вывода, либо оба эти ресурса. Эта функция аналогична функциям организации много-

[3] В действительности все они, кроме одного, являются библиотечными функциями.

поточной обработки в большинстве версий UNIX, но в Linux каждый поток выполнения представлен в виде полноценного процесса, а не специализированным "потоковым" объектом.

> 🕮 Подробная информация о начальной загрузке и демоне `init` приведена в главе 2.

Когда система загружается, ядро самостоятельно запускает несколько процессов. Наиболее важный из них — это демон `init`, идентификатор которого всегда равен 1. Демон `init` отвечает за выполнение сценариев запуска системы. Все процессы, кроме тех, что создаются ядром, являются потомками демона `init`.

Демон `init` играет и другую важную роль в управлении процессами. Завершающийся процесс вызывает функцию `_exit()`, чтобы уведомить ядро о своей готовности прекратить работу. В качестве параметра функции `_exit()` передается код завершения — целое число, обозначающее причину останова процесса. Нулевой код свидетельствует об успешном завершении процесса.

В Linux требуется, чтобы, прежде чем процесс окончательно исчезнет, его удаление было подтверждено родительским процессом с помощью системного вызова `wait`. Этот вызов возвращает код завершения потомка (или сообщает о причине его уничтожения, если завершение не было естественным) и, в случае необходимости, статистику использования ресурсов.

Описанный механизм работает нормально, если родительский процесс завершается позже порожденных им процессов и выполняет системные вызовы `wait` для их уничтожения. Если же родительский процесс завершается раньше срока, то ядро "понимает", что вызова `wait` не последует, и переназначает все "осиротевшие" процессы демону `init`. Он берет их под свой контроль, осуществляя для каждого для них вызов wait.

4.3. Сигналы

Сигналы — это запросы на прерывание, реализуемые на уровне процессов. Определено свыше тридцати различных сигналов, и они находят самое разное применение.

- Сигналы могут посылаться между процессами как средство коммуникации.
- Сигналы могут посылаться драйвером терминала для уничтожения или приостановки процессов, когда пользователь нажимает специальные сочетания клавиш, такие как <Ctrl+C> или <Ctrl+Z>[4].
- Сигналы могут посылаться в самых разных целях пользователем или администратором с помощью команды `kill`.
- Сигналы могут посылаться ядром, когда процесс выполняет нелегальную инструкцию, например деление на нуль.
- Сигналы могут посылаться ядром для уведомления процесса о "представляющем интерес" событии, таком как прекращении дочернего процесса или доступности данных в канале ввода-вывода.

> 🕮 Дамп памяти — это файл, содержащий образ памяти процесса. Его можно использовать для отладки.

[4] Функции, связанные с этими сочетаниями клавиш, могут назначаться другим клавишам с помощью команды `stty`, но на практике такое встречается очень редко. В этой главе мы подразумеваем, что с данными клавишами связаны их стандартные функции. Дополнительную информацию можно получить в разделе 7.11.

Когда поступает сигнал, возможен один из двух вариантов развития событий. Если процесс назначил сигналу подпрограмму обработки, то после вызова ей предоставляется информация о контексте, в котором был сгенерирован сигнал. В противном случае ядро выполняет от имени процесса действия, заданные по умолчанию. Эти действия зависят от сигнала. Многие сигналы приводят к завершению процесса, а в некоторых случаях при этом еще и создается дамп памяти.

Процедура вызова обработчика называется *перехватом* сигнала. Когда выполнение обработчика завершается, процесс возобновляется с той точки, где был получен сигнал.

Для того чтобы определенные сигналы не поступали в программу, нужно задать их игнорирование или блокирование. Игнорируемый сигнал просто пропускается и не влияет на работу процесса. Блокируемый сигнал ставится в очередь на обработку, но ядро не требует от процесса никаких действий до явного разблокирования сигнала. Обработчик вызывается для разблокированного сигнала только один раз, даже если в течение периода блокировки поступило несколько аналогичных сигналов.

В табл. 4.1 перечислены сигналы, которые должны быть известны любому системному администратору. Традиционно имена сигналов записываются прописными буквами. Иногда к именам добавляется префикс `SIG` (например, `SIGHUP`).

Таблица 4.1. Сигналы, которые должен знать каждый администратор

#	Имя	Описание	Реакция по умолчанию	Перехватывается?	Блокируется?	Дамп памяти?
1	`HUP`	Отбой	Завершение	Да	Да	Нет
2	`INT`	Прерывание	Завершение	Да	Да	Нет
3	`QUIT`	Выход	Завершение	Да	Да	Да
9	`KILL`	Уничтожение	Завершение	Нет	Нет	Нет
[a]	`BUS`	Ошибка на шине	Завершение	Да	Да	Да
11	`SEGV`	Ошибка сегментации	Завершение	Да	Да	Да
15	`TERM`	Запрос на завершение	Завершение	Да	Да	Нет
[a]	`STOP`	Останов	Останов	Нет	Нет	Нет
[a]	`TSTP`	Сигнал останова, посылаемый с клавиатуры	Останов	Да	Да	Нет
[a]	`CONT`	Продолжение после останова	Игнорируется	Да	Нет	Нет
[a]	`WINCH`	Изменение окна	Игнорируется	Да	Да	Нет
[a]	`USR1`	Определяется пользователем	Завершение	Да	Да	Нет
[a]	`USR2`	Определяется пользователем	Завершение	Да	Да	Нет

[a] Зависит от типа аппаратуры (см. раздел man 7 signal интерактивного руководства).

Существуют и другие сигналы, не показанные в табл. 4.1; большинство из них сообщает о всяких загадочных ошибках, например “неверная инструкция”. По умолчанию такие сигналы, как правило, приводят к завершению программы и созданию дампа памяти. Перехват и блокирование сигналов обычно разрешены, так как есть достаточно “умные” программы, старающиеся устранять последствия ошибок.

Сигналы BUS и SEGV также посылаются при возникновении ошибок. Мы включили их в таблицу, поскольку они чрезвычайно распространены: в 99% случаев программа аварийно завершается именно из-за них. Сами по себе эти сигналы не имеют диагностической ценности. Они лишь указывают на факт неправильного обращения к памяти[5].

Сигналы KILL и STOP нельзя ни перехватить, ни заблокировать, ни проигнорировать. Сигнал KILL приводит к уничтожению процесса, которому он посылается, а сигнал STOP приостанавливает выполнение процесса до получения сигнала CONT. Сигнал CONT можно перехватить и проигнорировать, но нельзя заблокировать.

Сигнал TSTP представляет собой более "гибкую" версию сигнала STOP. Проще всего описать его как запрос на останов. Он генерируется драйвером терминала при нажатии пользователем клавиш <Ctrl+Z>. Программы, перехватывающие этот сигнал, обычно выполняют операции очистки, а затем посылают сами себе сигнал STOP. С другой стороны, программы могут игнорировать сигнал TSTP, чтобы их нельзя было остановить командой с клавиатуры.

Эмуляторы терминалов посылают сигнал WINCH, когда происходит изменение их конфигурационных параметров (например, числа строк на виртуальном терминале). Это позволяет программам, которые взаимодействуют с эмуляторами, таким как текстовые редакторы, автоматически переконфигурировать себя в ответ на изменения. Если размер окна не удается правильно изменить, убедитесь в том, что сигнал WINCH генерируется и корректно обрабатывается[6].

Хотя назначение сигналов KILL, INT, TERM, HUP и QUIT может показаться одинаковым, в действительности они совершенно различны.

- Сигнал KILL не блокируется и приводит к безусловному завершению процесса на уровне ядра. По сути, процесс не успевает даже принять этот сигнал.
- Сигнал INT посылается драйвером терминала при нажатии пользователем клавиш <Ctrl+C> и служит запросом на завершение текущей операции. Перехватив этот сигнал, простые программы должны завершить работу или позволить уничтожить себя стандартному обработчику сигнала. Программы, в которых есть режим командной строки, должны прекратить текущую операцию, выполнить очистку и снова перейти в режим ожидания.
- Сигнал TERM представляет собой запрос на завершение программы. Предполагается, что процесс, получивший этот сигнал, осуществляет очистку и завершается.

[5] Точнее говоря, ошибки на шине возникают в результате нарушения различных требований синхронизации или использования бессмысленных адресов. Нарушения правил сегментирования представляют собой нарушения правил защиты, таких как попытки выполнения записи в части адресного пространства, доступные только для чтения.

[6] На практике это может оказаться не такой уж простой задачей. И эмулятор терминала (например, xterm), и драйвер терминала, и команды пользовательского уровня — все они могут вносить свой вклад в распространение сигнала SIGWINCH. Распространенные проблемы — отправка сигнала только фоновому процессу терминала (а не всем связанным с ним процессам) без рассылки по сети уведомления об изменении размера удаленным компьютерам. Такие протоколы, как TELNET и SSH, явным образом распознают изменения размера локального терминала и передают эту информацию удаленному хосту. Более простые протоколы (например, протоколы прямого последовательного подключения) лишены этой возможности.

- У сигнала HUP есть две распространенные интерпретации. Во-первых, многие демоны воспринимают его как команду сброса. Если демон способен повторно прочесть свой конфигурационный файл и адаптироваться к изменениям без перезапуска, сигнал HUP позволяет менять его поведение.

  Во-вторых, этот сигнал иногда генерируется драйвером терминала при попытке уничтожить связанные с терминалом процессы. В основном это поведение сохранилось со времен использования проводных соединений терминалов и модемов. Отсюда и название "отбой".

  Интерпретаторы семейства csh (tcsh и др.) обычно делают фоновые процессы невосприимчивыми к сигналу HUP, чтобы они могли продолжать свою работу, даже когда пользователь выходит из системы. Пользователи интерпретаторов семейства sh (ksh, bash и т.д.) могут эмулировать такое поведение с помощью команды nohup.
- Сигнал QUIT напоминает сигнал TERM, за исключением того, что по умолчанию стандартный обработчик создает дамп памяти.

Сигналы USR1 и USR2 не имеют стандартного назначения. Ими можно пользоваться в различных целях. Например, Web-сервер Apache интерпретирует сигнал USR1 как запрос на перезапуск.

4.4. Отправка сигналов: команды kill и killall

Команду kill чаще всего используют для уничтожения процессов. Эта команда может послать процессу любой сигнал, но по умолчанию это сигнал TERM. Команду kill могут выполнять как рядовые пользователи (для своих собственных процессов), так и пользователь root (для любого процесса). Она имеет следующий синтаксис:

```
kill [-сигнал] идентификатор
```

где *сигнал* — это номер или символическое имя посылаемого сигнала (см. табл. 4.1), а *идентификатор* — номер искомого процесса. *Идентификатор* -1 означает широковещательную передачу сигнала всем процессам, кроме демона init.

Команда kill без номера сигнала не гарантирует, что процесс будет уничтожен, поскольку сигнал TERM можно перехватывать, блокировать и игнорировать. Команда

```
kill -KILL pid
```

"гарантированно" уничтожает процесс, так как сигнал номер 9, KILL, не перехватывается. Мы написали "гарантированно" в кавычках, так как иногда процессы попадают в состояние, когда их нельзя завершить даже таким способом (обычно это связано с блокировкой ввода-вывода, например при остановке жесткого диска). Единственный выход в такой ситуации — перезагрузка.

Большинство интерпретаторов имеет собственную встроенную реализацию команды kill, которая подчиняется описанному выше синтаксису. Согласно информации, приведенной на man-странице системной команды kill, имени или номеру сигнала должен предшествовать флаг -s (например, kill -s HUP). Однако не все интерпретаторы понимают такой синтаксис, поэтому мы советуем придерживаться традиционного син-

таксиса (без флага -s), так как команда kill прекрасно работает и в отсутствие флага. В этом случае различия между версиями команды станут несущественными.

Если идентификатор процесса, которому нужно послать сигнал, неизвестен, воспользуйтесь командой ps (описана в разделе 4.7). Другой вариант — это команда killall, самостоятельно выполняющая такой поиск. Например, следующая команда заставляет демон xinetd обновить свою конфигурацию:

```
$ sudo killall -USR1 xinetd
```

Следует отметить, что при наличии нескольких процессов, соответствующих указанной в команде строке, killall отправит сигналы всем этим процессам.

Похожая возможность есть и у обычной команды kill, но команда killall лучше сопоставляет имена идентификаторам.

4.5. Состояния процесса

Факт существования процесса не дает ему автоматически права на получение доступа к центральному процессору. Необходимо помнить о четырех состояниях выполнения процесса (табл. 4.2).

Таблица 4.2. Состояния процесса

Состояние	Описание
Выполнение	Процесс можно выполнять
Ожидание	Процесс ждет выделения ресурса
Зомби	Процесс пытается завершиться
Останов	Процесс приостановлен (не имеет разрешения на выполнение)

Выполняемый процесс получил все необходимые ресурсы и ждет, пока системный планировщик предоставит ему доступ к центральному процессору. Если процесс осуществляет системный вызов, который нельзя завершить немедленно (например, чтение части файла), Linux переводит его в состояние ожидания.

Ожидающий процесс ждет наступления определенного события. Интерактивные интерпретаторы команд и системные демоны проводят в этом состоянии большую часть времени, ожидая поступления данных с терминала или из сетевого соединения. Поскольку ожидающий процесс фактически блокирован, доступ к центральному процессору будет предоставлен ему только в случае получения сигнала.

Некоторые операции переводят процесс в состояние непрерывного ожидания. Обычно это состояние является временным и не отображается в выводе команды ps (о чем свидетельствует флаг D в столбце STAT (см. раздел 4.7). Однако некоторые нестандартные ситуации могут приводить к сохранению этого состояния надолго. Наиболее распространенная причина возникновения такой ситуации — наличие проблем сервера в файловой системе NFS смонтированной с применением опции “hard”. Поскольку процессы в состоянии непрерывного ожидания не могут быть активизированы даже для реакции на сигнал, их нельзя уничтожить. Чтобы избавиться от них, необходимо устранить породившую их проблему или перезагрузить систему.

Зомби — это процесс, который закончил выполняться, но информация об этом еще не поступила родительскому процессу. При обнаружении зомби с помощью команды `ps` проверьте их атрибуты PPID, чтобы выяснить происхождение этих процессов.

Остановленному процессу временно запрещено выполняться. Процессы останавливаются при получении сигнала `STOP` или `TSTP` и возобновляют работу по сигналу `CONT`. Это состояние аналогично ожиданию, но выход из него возможен только с помощью другого процесса.

4.6. Изменение приоритета выполнения: команды `nice` и `renice`

Фактор "уступчивости" — это число, по которому ядро определяет свою политику в отношении процессов, конкурирующих за право доступа к центральному процессору. Чем выше фактор уступчивости, тем ниже приоритет процесса, и наоборот, отсюда название термина. Низкое или отрицательное значение означает использование высокого приоритета: процесс ведет себя не слишком уступчиво. Диапазон допустимых значений — от -20 до +19.

Если пользователь не предпринимает специальных мер, то дочерний процесс наследует приоритет своего родительского процесса. Владелец процесса может увеличить фактор уступчивости, но не может уменьшить его, даже чтобы вернуться к стандартному значению. Это не позволяет процессам с низким приоритетом порождать высокоприоритетных потомков. Суперпользователь может устанавливать произвольные значения фактора уступчивости.

В настоящее время администраторам редко приходится определять приоритеты вручную. Когда операционные системы работали на маломощных компьютерах 70–80 гг., на производительность больше всего влияло то, какой процесс занимал основную часть времени центрального процессора. Сегодня, когда на рабочих столах стоят намного более быстродействующие компьютеры, системный планировщик, как правило, обслуживает все процессы весьма оперативно. Добавление классов планирования предоставляет разработчикам дополнительные средства управления в тех случаях, когда важна низкая ответная латентность.

К сожалению, уровень производительности подсистемы ввода-вывода растет не так стремительно, как быстродействие центральных процессоров, поэтому жесткие диски стали основным узким местом в большинстве операционных систем. Фактор уступчивости никак не влияет на подсистемы управления памятью и вводом-выводом, поэтому даже низкоприоритетный процесс способен монополизировать эти ресурсы или захватить непропорционально большую их часть.

Фактор уступчивости можно установить при создании процесса. Это делается с помощью команды `nice`. Команда `renice` позволяет изменять приоритет выполняемого процесса. Первая из этих команд принимает в качестве аргумента строку запуска процесса, а вторая — идентификатор процесса либо имя пользователя. Важный нюанс: команда `renice` требует указания абсолютного приоритета, а команда `nice` — *относительного*, прибавляемого к текущему значению приоритета интерпретатора или вычитаемого из него.

Приведем примеры:

```
% nice -n 5 ~/bin/longtask      // Понижаем приоритет (увеличиваем фактор
                                // уступчивости) на 5
% sudo renice -5 8829           // Задаем фактор уступчивости равным -5
% sudo renice 5 -u boggs        // Задаем фактор уступчивости процессов
                                // пользователя "boggs" равным 5
```

Существует также версия команды `nice`, встроенная в интерпретатор `csh` и ряд других популярных интерпретаторов (но не в `bash`). Если не указать полное имя команды, будет вызвана именно встроенная версия, а не системная. Это многих сбивает с толку, так как синтаксис команд различен: встроенная версия требует, чтобы изменение приоритета записывалось в формате *+приращение* или *-приращение*, а системная версия ожидает флаг `-n`, за которым следует значение приращения[7].

Самый распространенный из высокоприоритетных процессов в современных системах — `xntpd`, демон тактовой синхронизации. Поскольку для него быстрый доступ к центральному процессору имеет очень большое значение, этому демону обычно назначается фактор уступчивости, на 12 позиций ниже стандартного уровня.

Если какой-либо процесс займет столько времени центрального процессора, что показатель средней загруженности системы достигнет отметки 65, то перед выполнением команд, необходимых для исследования этой проблемы, понадобится запустить с помощью команды `nice` высокоприоритетный интерпретатор. В противном случае выполнение даже простейших команд может быть затруднено.

4.7. Текущий контроль процессов: команда ps

Команда `ps` — это основной инструмент, которым системный администратор пользуется для текущего контроля процессов. С ее помощью можно получить информацию об идентификаторах, приоритете и управляющем терминале того или иного процесса. Она также позволяет выяснить, какой объем памяти использует процесс, сколько времени центрального процессора заняло его выполнение, каково его текущее состояние (выполняется, остановлен, простаивает и т.д.). Процессы-зомби в листинге команды обозначаются как `<defunct>`.

Работа команды `ps` сильно различается в разных вариантах UNIX, а многие реализации команды стремительно усложнились за последние несколько лет. В Linux команда `ps` является настоящим хамелеоном и понимает наборы опций из целого ряда других систем. Есть специальная переменная среды, с помощью которой можно выбрать нужный набор.

Не пугайтесь всех этих сложностей: пусть они будут кошмаром разработчиков ядра, а не системных администраторов! Для повседневной работы достаточно знать лишь несколько наиболее важных опций команды `ps`.

Получить список всех процессов, выполняющихся в системе, можно с помощью команды `ps aux`.

Ниже показаны результаты ее работы (мы удалили столбец `START`, чтобы результаты поместились на странице).

[7] Хуже того: системная версия команды интерпретирует строку `nice -5` как положительное приращение, а встроенная версия интерпретирует эту же строку как отрицательное приращение.

```
$ ps aux
  USER    PID  %CPU  %MEM    VSZ  RSS   TTY   STAT    TIME   COMMAND
  root      1   0.1   0.2   3356  560   ?     S       0:00   init [5]
  root      2   0     0        0  0     ?     SN      0:00   [ksoftirqd/0]
  root      3   0     0        0  0     ?     S<      0:00   [events/0]
  root      4   0     0        0  0     ?     S<      0:00   [khelper]
  root      5   0     0        0  0     ?     S<      0:00   [kacpid]
  root     18   0     0        0  0     ?     S<      0:00   [kblockd/0]
  root     28   0     0        0  0     ?     S       0:00   [pdflush]
   ...
  root    196   0     0        0  0     ?     S       0:00   [kjournald]
  root   1050   0     0.1   2652  448   ?     S<s     0:00   udevd
  root   1472   0     0.3   3048  1008  ?     S<s     0:00   /sbin/dhclient -1
  root   1646   0     0.3   3012  1012  ?     S<s     0:00   /sbin/dhclient -1
  root   1733   0     0        0  0     ?     S       0:00   [kjournald]
  root   2124   0     0.3   3004  1008  ?     Ss      0:00   /sbin/dhclient -1
  root   2182   0     0.2   2264  596   ?     Ss      0:00   syslogd -m 0
  root   2186   0     0.1   2952  484   ?     Ss      0:00   klogd -x
   rpc   2207   0     0.2   2824  580   ?     Ss      0:00   portmap
rpcuser  2227   0     0.2   2100  760   ?     Ss      0:00   rpc.statd
  root   2260   0     0.4   5668  1084  ?     Ss      0:00   rpc.idmapd
  root   2336   0     0.2   3268  556   ?     Ss      0:00   /usr/sbin/acpid
  root   2348   0     0.8   9100  2108  ?     Ss      0:00   cupsd
  root   2384   0     0.6   4080  1660  ?     Ss      0:00   /usr/sbin/sshd
  root   2399   0     0.3   2780  828   ?     Ss      0:00   xinetd -stayalive
  root   2419   0     1.1   7776  3004  ?     Ss      0:00   sendmail: accepi
   ...
```

Команды, имена которых заключены в квадратные скобки, в действительности являются не командами, а потоками ядра, запланированными в качестве процессов. Описание полей приведено в табл. 4.3.

Еще один полезный набор аргументов команды `ps` — `lax`, предоставляющий дополнительную техническую информацию. В данном случае команда выдает более специализированную информацию и выполняется быстрее, так как не сопоставляет идентификаторы процессов именам пользователей. Это может оказаться весьма важным фактором, если система уже перегружена каким-то процессом.

Таблица 4.3. Пояснения к выходной информации команды `ps aux`

Поле	Содержимое
`USER`	Имя владельца процесса
`PID`	Идентификатор процесса
`%CPU`	Доля времени центрального процессора (в процентах), выделенная процессу
`%MEM`	Часть реальной памяти (в процентах), используемая процессом
`VSZ`	Виртуальный размер процесса
`RSS`	Размер резидентного набора (количество страниц памяти)
`TTY`	Идентификатор управляющего терминала
`STAT`	Текущий статус процесса: `R` — выполняется, `D` — в режиме непрерывного ожидания, `S` — неактивен (< 20 с), `T` — приостановлен, `Z` — зомби.

Окончание табл. 4.3

Поле	Содержимое
	Дополнительные флаги: W — процесс выгружен на диск, < — процесс имеет повышенный приоритет, N — процесс имеет пониженный приоритет, L — некоторые страницы блокированы в оперативной памяти s — процесс является основным процессом сеанса
START	Время запуска процесса
TIME	Количество времени центрального процессора, затраченное на выполнение процесса
COMMAND	Имя и аргументы команды[a]

[a] Программы могут модифицировать эту информацию, так что она не всегда точно представляет реальную командную строку.

Ниже в сокращенном виде показаны результаты работы команды `ps lax`. Обратите внимание на дополнительные поля `PPID` (идентификатор родительского процесса), `NI` (фактор уступчивости) и `WCHAN` (ресурс, которого ожидает процесс):

```
$ ps lax
  F  UID   PID  PPID  PRI  NI    VSZ    RSS  WCHAN     STAT   TIME   COMMAND
  4    0     1     0   16   0   3356    560  select    S      0:00   init [5]
  1    0     2     1   34  19      0      0  ksofti    SN     0:00   [ksoftirqd/0
  1    0     3     1    5 -10      0      0  worker    S<     0:00   [events/0]
  1    0     4     3    5 -10      0      0  worker    S<     0:00   [khelper]
  5    0  2186     1   16   0   2952    484  syslog    Ss     0:00   klogd -x
  5   32  2207     1   15   0   2824    580  -         Ss     0:00   portmap
  5   29  2227     1   18   0   2100    760  select    Ss     0:00   rpc.statd
  1    0  2260     1   16   0   5668   1084  -         Ss     0:00   rpc.idmapd
  1    0  2336     1   21   0   3268    556  select    Ss     0:00   acpid
  5    0  2384     1   17   0   4080   1660  select    Ss     0:00   sshd
  1    0  2399     1   15   0   2780    828  select    Ss     0:00   xinetd -sta
  5    0  2419     1   16   0   7776   3004  select    Ss     0:00   sendmail: a
```

4.8. Улучшенный текущий контроль процессов: команда top

Команда `ps` позволяет сделать только разовый, моментальный "снимок" системы, но получить полную картину всего происходящего в ней довольно сложно. Для этого существует команда top, которая выдает регулярно обновляемую сводку активных процессов и используемых ими ресурсов.

Вот образец результатов ее работы:

```
top - 16:37:08 up  1:42,  2 users,  load average: 0.01, 0.02, 0.06
Tasks:  76 total,   1 running,  74 sleeping,   1 stopped,   0 zombie
Cpu(s):  1.1% us,  6.3% sy,  0.6% ni, 88.6% id,  2.1% wa,  0.1% hi,  1.3% si
Mem:    256044k  total, 254980k  used,   1064k  free,    15944k  buffers
Swap:   524280k  total,      0k  used, 524280k  free,   153192k  cached
```

```
  PID  USER    PR   NI   VIRT   RES  SHR    S   %CPU   %MEM  TIME+     COMMAND
 3175  root    15    0  35436   12m  4896   S    4.0    5.2  01:41.9   X
 3421  root    25   10  29916   15m  9808   S    2.0    6.2  01:10.5   rhn-applet-gui
    1  root    16    0   3356   560  480    S    0.0    0.2  00:00.9   init
    2  root    34   19      0     0  0      S    0.0    0    00:00.0   ksoftirqd/0
    3  root     5  -10      0     0  0      S    0.0    0    00:00.7   events/0
    4  root     5  -10      0     0  0      S    0.0    0    00:00.0   khelper
    5  root    15  -10      0     0  0      S    0.0    0    00:00.0   kacpid
   18  root     5  -10      0     0  0      S    0.0    0    00:00.0   kblockd/0
   28  root    15    0      0     0  0      S    0.0    0    00:00.0   pdflush
   29  root    15    0      0     0  0      S    0.0    0    00:00.3   pdflush
   31  root    13  -10      0     0  0      S    0.0    0    00:00.0   aio/0
   19  root    15    0      0     0  0      S    0.0    0    00:00.0   khubd
   30  root    15    0      0     0  0      S    0.0    0    00:00.2   kswapd0
  187  root     6  -10      0     0  0      S    0      0    00:00.0   kmirrord/0
  196  root    15    0      0     0  0      S    0      0    00:01.3   kjournald
  ...
```

По умолчанию эта информация обновляется каждые десять секунд. Наиболее активные процессы отображаются первыми. Команда `top` позволяет также посылать процессам сигналы и использовать команду `renice`, чтобы пользователь мог наблюдать за тем, как его действия влияют на общее состояние системы.

Пользователь root может запустить команду `top` с опцией `-q`, чтобы обеспечить ей максимально возможный приоритет. Это очень удобно, если нужно обнаружить процесс, который существенно замедлил работу системы.

4.9. Файловая система /proc

Версии команд `ps` и `top`, используемые в Linux, считывают информацию о состоянии процессов из каталога `/proc` — псевдо-файловой системы, в который ядро помещает большой объем интересной информации о состоянии системы. Несмотря на имя `/proc` (и имя базового типа файловой системы — “proc”), хранящаяся в этом каталоге информация не ограничивается одними лишь процессами — здесь хранится вся информация о состоянии и статистические сведения, генерируемые ядром. Некоторые параметры можно даже изменять, записывая новые значения в соответствующий файл каталога `/proc` — ряд примеров приведен в разделе 28.4.

Хотя некоторую информацию проще получать с помощью таких интерфейсных команд как `vmstat` и `ps`, некоторую менее популярную информацию придется считывать непосредственно из каталога `/proc`. Поэтому имеет смысл просмотреть его, чтобы ознакомиться со всеми помещенными в него файлами. Команда `man proc` позволяет ознакомиться с рядом полезных советов и приемов.

Поскольку ядро создает содержимое файлов каталога `/proc` на лету (во время их считывания), большинство из них выглядит пустыми при их открытии с помощью команды `ls -l`. Для просмотра действительного содержимого этих файлов придется прибегнуть к командам `cat` или `more`. Но будьте осторожны: некоторые файлы содержат либо ссылаются на двоичные данные, непосредственный просмотр которых может поставить в тупик эмулятор терминала.

Информация, относящаяся к конкретным процессам, распределена по подкаталогам, названным по идентификаторам процессов. Например, каталог `/proc/1` всегда содержит информацию о демоне `init`. Наиболее полезные файлы с информацией о процессах перечислены в табл. 4.4 .

Таблица 4.4. Файлы с информацией о процессах каталога /proc (нумерованные подкаталоги)

Файл	Содержимое
cmd	Команда или программа, выполняемая процессом
cmdline[a]	Полная командная строка процесса (разделенная нулями)
cwd	Символьная ссылка на текущий каталог процесса
environ	Переменные среды процесса (разделенные нулями)
exe	Символьная ссылка на файл, который должен выполняться
fd	Подкаталог, содержащий ссылки на дескрипторов каждого из отрытых файлов
maps	Информация отображения памяти (сегменты совместного использования, библиотеки и т.п.)
root	Символьная ссылка на корневой каталог процесса (определенный командой chroot)
stat	Информация об общем состоянии процесса (для ее получения лучше всего использовать команду ps)
statm	Информация об использовании памяти

[a] Может быть недоступна, если запись информации о процессе выполняется из памяти.

Отдельные компоненты внутри файлов cmdline и environ разделены символами нуля, а не символами перевода строки. Чтобы их содержимое было более читабельным, его можно фильтровать с помощью команды tr "\000" "\n".

В подкаталоге fd открытые файлы представлены символьными ссылками. Дескрипторы файлов, которые связаны с каналами или сетевыми сокетами, не имеют связанных с ними имен файлов. Вместо этого в качестве целевой ссылки ядро предоставляет обобщенное описание.

Файл maps полезен при определении библиотек, с которыми связана или от которых зависит та или иная программа.

4.10. Отслеживание сигналов и системных вызовов: команда strace

В традиционной системе UNIX определение действий, действительно выполняемых данным процессом, может быть затруднительным. Умозаключение приходится делать на основе косвенных данных, полученных от файловой системы и с помощью таких средств, как программа ps. В отличие от UNIX, Linux позволяет непосредственно следить за процессом с помощью команды strace, которая отображает каждый системный вызов, выполняемый процессом, и каждый получаемый им сигнал. Эту команду можно даже связать с выполняемым процессом, последить за ним в течение некоторого времени, а затем отключиться от процесса, не прерывая его[8].

Хотя системные вызова выполняются на сравнительно низком уровне абстракции, обычно вывод команды strace позволяет получить достаточно полную информацию об

[8] По крайней мере, как правило. В некоторых случаях, команда strace может прерывать системные вызовы. В этих случаях отслеживаемый процесс должен быть подготовлен для повторного запуска этих вызовов. Таково стандартное правило выполнения программ в среде UNIX, но оно не всегда соблюдается.

активности процесса. Например, следующий журнал был получен в результате выполнения команды strace применительно к активной копии команды top:

```
$ sudo strace -p 5810
gettimeofday({1116193814, 213881}, {300, 0})                = 0
open("/proc", O_RDONLY|O_NONBLOCK|O_LARGEFILE|O_DIRECTORY)  = 7
fstat64(7, {st_mode=S_IFDIR|0555, st_size=0, ...})          = 0
fcntl64(7, F_SETFD, FD_CLOEXEC)                             = 0
getdents64(7, /* 36 entries */, 1024)                       = 1016
getdents64(7, /* 39 entries */, 1024)                       = 1016
stat64("/proc/1", {st_mode=S_IFDIR|0555, st_size=0, ...})   = 0
open("/proc/1/stat", O_RDONLY)                              = 8
read(8, "1 (init) S 0 0 0 0 -1 4194560 73"..., 1023)        = 191
close(8)                                                    = 0
...
```

Команда strace не только отображает имя каждого системного вызова, выполненного процессом, но и раскрывает аргументы и отображает результирующий код, возвращенный ядром.

В этом примере команда top начинает свою работу с проверки текущего значения времени. Затем она открывает каталог /proc и считывает его содержимое, тем самым, получая список процессов, выполняемых в текущий момент. Команда top обращается к статической копии каталога, представляющей процесс init, а затем открывает /proc/1/stat, чтобы прочесть информацию о состоянии этого процесса.

4.11. Процессы, вышедшие из-под контроля

📖 Об обработке неуправляемыми процессами рассказывается также в разделе 25.4.

Неуправляемые процессы бывают двух видов: пользовательские, потребляющие слишком много системных ресурсов (например, времени центрального процессора или дискового пространства), и системные, которые внезапно выходят из-под контроля и начинают вести себя непредсказуемо. Процесс первого вида может быть вполне работоспособным, просто он некорректно обращается с ресурсами. В то же время системные процессы всегда должны подчиняться определенным правилам.

Процессы, занимающие чересчур много времени центрального процессора, можно выявить, проанализировав результаты работы команды ps или top. Если очевидно, что пользовательский процесс потребляет больше ресурсов, чем ему действительно необходимо, его нужно исследовать. На сервере или в системе совместного использования в первую очередь следует связаться с владельцем процесса и спросить, что происходит. Если это невозможно, придется действовать на свой страх и риск. Обычно системный администратор старается не заходить в каталоги пользователей, но это допускается, если нужно изучить исходный текст неуправляемого процесса и выяснить, что же он делает.

Это может понадобиться по двум причинам. Во-первых, процесс может быть вполне законным и действительно важным для пользователя. Нельзя же уничтожать процесс только из-за того, что он занимает много времени центрального процессора.

Во-вторых, процесс может нести угрозу системе и выполнять разрушительные действия. В этом случае нужно точно узнать, что он натворил (например, взломал пароли), и устранить повреждения.

Если причину появления неуправляемого процесса не удается определить, приостановите его сигналом STOP и отправьте владельцу по электронной почте сообщение,

поясняющее причину такого шага. Процесс можно впоследствии возобновить сигналом CONT. Помните о том, что некоторые процессы перестают нормально функционировать после долгого простоя, поэтому описанный подход не всегда приемлем. Например, процесс осуществлял обмен данными через сетевое соединение, а после "пробуждения" обнаруживается, что соединение уже разорвано.

Если правильно работающий процесс требует слишком много ресурсов процессора, но это является обоснованным, необходимо с помощью команды renice понизить его приоритет и попросить владельца в будущем запускать процесс с более низким приоритетом.

Процессы, которые используют значительный объем физического ОЗУ, могут существенно снижать производительность системы. Размер памяти, занимаемый процессами, можно проверить с помощью команды top. Столбец VIRT содержит информацию об общем объеме виртуальной памяти, зарезервированной каждым процессом, а столбец RES — информацию о части этой памяти, которая в текущий момент времени записана в конкретные страницы памяти ("резидентный набор").

Оба эти значения могут отражать ресурсы совместного использования, такие как библиотеки, что теоретически может приводить к неправильной оценке реальной ситуации. Более точная информация о памяти, потребляемой процессом, содержится в столбце DATA, который по умолчанию не отображается. Чтобы добавить этот столбец к информации, отображаемой командой top, после ее запуска нажмите клавишу f и выберите столбец DATA из списка. Отображаемое в этом столбце значение представляет объем памяти, занимаемый сегментами данных и стека каждого процесса, поэтому оно сравнительно специфично для отдельных процессов (оно представляет собой остаток деления на объем сегментов памяти совместного использования). По нему можно судить как об относительном увеличении объема используемой памяти, так и об абсолютном ее объеме.

Результаты работы неуправляемых процессов могут заполнить всю файловую систему и вызвать, таким образом, бесчисленные проблемы. При переполнении файловой системы на экран выдается множество сообщений, и попытки записать что-либо на диск тоже повлекут за собой появление сообщений об ошибках.

Первое, что нужно сделать в подобной ситуации, — остановить процесс, который переполняет диск. Если на диске было достаточно свободного пространства, то в случае его внезапного заполнения можете быть уверены: тут что-то неладно. Не существует средства, аналогичного команде top, которое сообщило бы о том, кто активнее всего использует дисковое пространство. Но есть утилиты, позволяющие получить список открытых файлов и работающих с ними процессов (в частности, утилиты fuser и lsof, описанные в разделе 5.2).

Можно приостановить все подозрительные процессы, пока не обнаружится источник проблем. Выявив "нарушителя", удалите все созданные им файлы и не забудьте возобновить "дисциплинированные" процессы.

Старый и хорошо известный трюк злоумышленников — запуск бесконечного цикла в интерпретаторе команд для создания непустого дерева каталогов:

```
while 1
       mkdir adir
       cd adir
       touch afile
end
```

Такое иногда происходит, если злоумышленник проник через общедоступный компьютер, который по недосмотру был оставлен подключенным к системе. Созданное дерево каталогов не занимает много места на диске, просто переполняется таблица индексных дескрипторов файловой системы, и другие пользователи теряют возможность создавать новые файлы. Единственное, что можно сделать в подобной ситуации, — устранить последствия и предупредить пользователей о необходимости защиты своих учетных записей. Поскольку дерево каталогов, оставленное таким маленьким программным "перлом", может оказаться слишком большим для команды `rm -r`, придется написать сценарий для того, чтобы спуститься по дереву вниз, а затем, возвращаясь, удалить все каталоги.

Если проблема возникла в каталоге `/tmp`, а он был смонтирован как отдельная файловая система, то вместо того чтобы удалять файлы, можно проинициализировать каталог `/tmp` с помощью команды `mkfs`. Подробнее об управлении файловыми системами рассказывается в главе 7.

4.12. Рекомендуемая литература

- Bovet, Daniel P. and Marko Cesati. *Understanding the Linux Kernel (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2006.
- Роберт Лав, *Разработка ядра Linux, 2-е издание* (пер. с англ., ИД "Вильямс", 2006 г.)

4.13. Упражнения

4.1. Объясните, какая связь существует между значением UID исполняемого файла и реальным и эффективным идентификаторами выполняемого процесса. Каково назначение эффективного идентификатора процесса, помимо контроля доступа к файлам?

4.2. Предположим, что кто-то из пользователей системы запустил длительный и ресурсоемкий процесс.

а) Как можно было бы распознать процесс, который узурпирует ресурсы?

б) Вы подозреваете, что этот процесс необходим пользователю и не должен быть уничтожен. С помощью каких команд можно приостановить процесс на время проверки?

в) Впоследствии выясняется, что процесс запущен начальником и должен быть продолжен. С помощью каких команд можно возобновить выполнение задания?

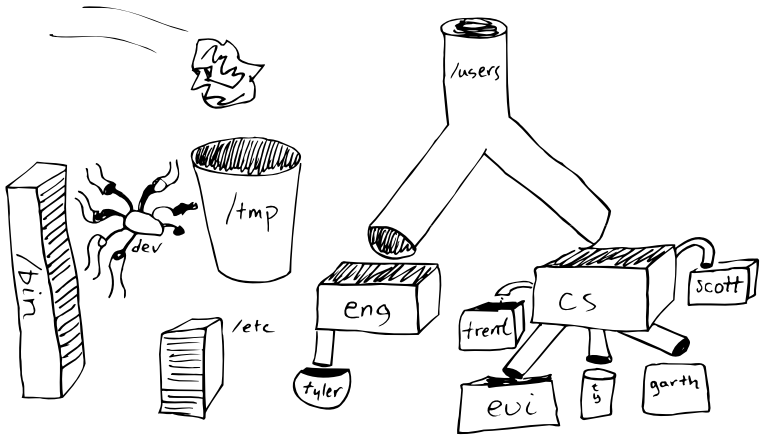
г) Теперь предположим, что процесс должен быть уничтожен. Какой сигнал нужно ему послать и почему? Что если необходимо гарантированно уничтожить процесс?

4.3. Найдите программу, в которой существует "утечка памяти" (при необходимости напишите собственную программу). Проследите за работой программы с помощью команды `ps` или `top`, обращая внимание на показатели использования памяти.

☆ 4.4. Напишите простой Perl-сценарий обработки вывода команды `ps` для определения суммарного значения VSZ и RSS процесса, выполняющегося в системе. Как эти показатели соотносятся с реальным объемом физической памяти и размером раздела подкачки?

Глава 5

Файловая система



Ответьте, не раздумывая, на вопрос: что из нижеперечисленного можно считать элементами файловой системы?

- Процессы
- Последовательные порты
- Структуры данных ядра и параметры настройки
- Каналы межзадачного взаимодействия

Если речь идет о Linux, ответом будет: все вышеперечисленное. Ну и, конечно же, в файловую систему входят собственно файлы[1].

Хотя основным назначением файловой системы является упорядочение хранимых ресурсов системы (т.е. файлов), программистам не хотелось каждый раз заново изобретать колесо при управлении объектами других типов. Очень часто оказывалось удобным представлять такие объекты в виде элементов файловой системы. Подобный унифицированный подход имеет как преимущества (единый программный интерфейс, легкий доступ из интерпретатора команд), так и недостатки (реализация файловых систем по методу доктора Франкенштейна). Но независимо от того, нравится он вам или нет, именно такой подход применяется в UNIX (а значит, и в Linux).

Файловую систему можно представить состоящей из четырех основных компонентов:

- пространство имен — методы именования объектов и организации их в виде единой иерархии;

[1] Точнее говоря, эти элементы *представлены* внутри файловой системы. В большинстве случаев файловая система служит "местом встречи" клиентов с драйверами и серверами, которые им требуются.

- API[2] — набор системных вызовов для перемещения между объектами и управления ими;
- модель безопасности — схема защиты, сокрытия и совместного использования объектов;
- реализация — программный код, который связывает логические модели с дисковой подсистемой.

Сетевая файловая система, NFS, рассматривается в главе 16.

В Linux определен абстрактный интерфейс уровня ядра, позволяющий работать с различными файловыми системами. Одни части файлового дерева обрабатываются традиционной дисковой подсистемой, другие — управляются специальными драйверами ядра. Например, за работу сетевых файловых систем (NFS) отвечает драйвер, который перенаправляет запросы серверу, расположенному на другом компьютере.

К сожалению, концептуальные границы нечетко очерчены, поэтому имеется слишком много "особых" случаев. К примеру, файлы устройств позволяют программам взаимодействовать с драйверами ядра. Они не являются файлами данных, но обрабатываются базовым драйвером файловой системы, а их характеристики записываются на диск. Наверное, имело бы смысл переписать код файловой системы с учетом опыта последнего десятилетия.

Информацию о файловой системе ReiserFS можно найти на Web-сайте `www.namesys.com`.

Другим усложняющим (но, в конечном счете, полезным) фактором является то, что в Linux поддерживается несколько типов дисковых файловых систем. Среди наиболее перспективных на данный момент файловых систем можно назвать ext3fs, которая во многих дистрибутивах является базовой, а также ReiserFS, JFS компании IBM и XFS компании SGI. Предшественница ext3fs — ext2fs — также продолжает поддерживаться всеми дистрибутивами и, судя по всему, будет поддерживаться еще долгое время.

Есть много реализаций файловых систем с другой семантикой, таких как FAT и NTFS (используются в Microsoft Windows) и 9660 (применяется на компакт-дисках). Linux поддерживает больше файловых систем, чем любая другая разновидность UNIX. Столь широкий выбор упрощает обмен файлами с другими системами.

Организация файловой системы — очень обширная тема, которую мы рассмотрим с нескольких точек зрения. В этой главе рассказывается, где обычно хранятся файлы, и описываются характеристики файлов, а также рассматриваются ключевые команды, позволяющие задавать атрибуты файлов. В главе 7 структура файловой системы описывается на более низком уровне. В главе 16 рассматриваются средства совместного доступа к файлам, столь часто используемые в Linux. Средства совместного доступа к файлам Microsoft Windows, используемые в Linux, рассматриваются в главе 26.

При столь большом разнообразии реализаций файловых систем, доступных в Linux, может показаться странным, что данная глава посвящена всего лишь одной файловой системе. Однако на отдельных реализациях можно специально не останавливаться, поскольку в большинстве современных файловых систем либо предприняты попытки обеспечить более эффективное и надежное применение традиционных функциональных возможностей, либо дополнительные функциональные возможности реализованы в ка-

[2] Программный интерфейс приложений (Application Programming Interface, API) — общий термин для описания набора подпрограмм, организованных в виде библиотеки или совокупности библиотек и доступных программистам.

честве отдельного слоя, работающего поверх семантики стандартной файловой системы (некоторые файловые системы реализуют оба названных направления). Как бы то ни было, слишком большое число программных пакетов ориентировано на модель, которая описана в этой главе, чтобы ею можно пренебречь.

5.1. Имена файлов и каталогов

Файловая система — это единая иерархическая структура[3], которая начинается с каталога / и разветвляется, охватывая произвольное число подкаталогов. Каталог самого верхнего уровня называется корневым.

Цепочка имен каталогов, через которые необходимо пройти для доступа к заданному файлу, вместе с именем этого файла образуют *путь к файлу*. Путь может быть абсолютным (например, `/tmp/foo`) или относительным (например, `book4/filesystem`). Последние интерпретируются, начиная с текущего каталога. Возможно, многие считают, что текущий каталог задается интерпретатором команд. На самом деле текущий каталог есть у каждого процесса.

Термины *имя файла* и *путь* в той или иной степени являются взаимозаменяемыми (по крайней мере, так они трактуются в данной книге). Соответственно, имя файла бывает полным (абсолютный путь) или сокращенным (относительный путь).

Файловое дерево может иметь произвольную глубину, однако каждый компонент имени файла должен состоять не более чем из 255 символов, а в отдельном имени не должно быть более 4095 символов. Чтобы получить доступ к файлу, полное имя которого превышает эти ограничения, необходимо с помощью команды `cd` перейти в промежуточный каталог, а затем воспользоваться сокращенным именем[4].

На имена файлов и каталогов не налагаются никакие ограничения, за исключением того, что длина имени не должна превышать заданный предел и в имя нельзя включать символы косой черты и нулевые символы. В частности, допускаются имена, содержащие пробелы. К сожалению, по давней традиции аргументы командной строки в UNIX разделяются пробелами, поэтому старые программы интерпретируют пробелы в именах файлов как признак конца одного имени и начала другого.

Первоначально пробелы в именах файлов применялись в основном в файловых системах, используемых на компьютерах Mac и PC, но теперь они проникли и в культуру Linux и их можно встретить и некоторых стандартных программных пакетах. По этому поводу не может быть двух мнений: административные сценарии должны быть готовы к обработке пробелов в именах файлов.

Об этом следует помнить при написании любых программ, взаимодействующих с файловой системой. В интерпретаторе команд и его сценариях имена с пробелами необходимо просто заключать в двойные кавычки. Например, команда

```
$ less "My excellent file.txt"
```

будет воспринята как попытка передать программе `less` единый аргумент `My excellent file.txt`. Отдельные пробелы можно также предварять символом обратной косой чер-

[3] В этом проявляется отличие от Windows, где сохраняется концепция обособленных дисковых пространств имен.

[4] Здесь необходимо дать пояснения. Сама дисковая подсистема не налагает никаких ограничений на общую длину имени файла. Но системные вызовы, получающие доступ к файловой системе, не позволяют своим строковым аргументам иметь длину более чем 4095 символов.

ты. Функция завершения имен файлов, предоставляемая современными интерпретаторами (обычно она связана с клавишей <Tab>), позволяет это делать автоматически.

При написании сценариев полезным инструментом является опция `-print0` команды `find`. В сочетании с опцией `-0` команды `xargs` эта опция позволяет сочетанию команд `find/xargs` работать правильно независимо от наличия пробелов в именах файлов. Например, команда

```
$ find /home -size +1M -print0 | xargs -0 ls -l
```

выводит длинный `ls`-список всех файлов, хранящихся в логическом разделе `/home`, размер которых превышает один мегабайт.

5.2. Монтирование и размонтирование файловой системы

Файловое дерево формируется из отдельных частей, называемых файловыми системами, каждая из которых содержит корневой каталог и список его подкаталогов и файлов. Термин "файловая система" имеет, по сути, два значения. С одной стороны, это составная часть файлового дерева, а с другой — все файловое дерево и алгоритмы, с помощью которых Linux управляет им. Как правило, значение термина становится ясным из контекста.

Большинство файловых систем является разделами диска, но, как уже было сказано, файловая система может принять облик всего, что подчиняется определенным функциональным правилам: сетевых файловых систем, компонентов ядра, резидентных дисков и т.д. В Linux есть даже оригинальная файловая система, позволяющая монтировать отдельные файлы, как если бы они были физическими устройствами.

Файловые системы присоединяются к файловому дереву с помощью команды `mount`. Эта команда связывает каталог существующего файлового дерева, называемый точкой монтирования, с корневым каталогом новой файловой системы. На время монтирования доступ к прежнему содержимому точки монтирования становится невозможным. Впрочем, в большинстве случаев точка монтирования — это пустой каталог.

Например,

```
# mount /dev/hda4 /users
```

монтирует на устройстве `/dev/hda4` файловую систему `/users`. По окончании монтирования можно с помощью команды `ls /users` просмотреть содержимое файловой системы.

Список смонтированных пользователями файловых систем хранится в файле `/etc/fstab`. Благодаря этому возможны автоматическая проверка (`fsck -A`) и монтирование (`mount -a`) файловых систем на этапе начальной загрузки, а также выполнение коротких команд наподобие `mount /usr` (точное местонахождение монтируемой файловой системы ищется в файле `/etc/fstab`). Информация, содержащаяся в этом файле, служит документацией к схеме расположения файловых систем на диске. Подробнее файл `/etc/fstab` описан в разделе 7.4.

Файловые системы размонтируются командой `umount`. "Занятую" файловую систему размонтировать невозможно. В ней не должно быть ни открытых файлов, ни текущих каталогов выполняющихся процессов. Если размонтируемая файловая система содержит исполняемые программы, они не должны быть запущены.

Версии ядра Linux 2.4.11 и последующие определяют опцию "ленивого" размонтирования (вызываемую командой `umount -l`), которая удаляет файловую систему из иерархии имен, но в действительности не размонтирует ее до тех пор, пока все существующие файловые ссылки не будут закрыты. Полезность этой опции достаточно спорна. Во-первых, нет никакой гарантии, что существующие ссылки когда-либо будут закрыты сами по себе, кроме того, "наполовину размонтированное" состояние файловой системы может нарушать семантическую целостность для использующих ее программ. Они могут выполнять операции чтения и записи с существующими дескрипторами файлов, но не могут открывать новые файлы или выполнять какие-либо другие операции файловой системы.

Когда ядро сообщает, что размонтируемая файловая система занята, запустите команду `fuser`, с помощью которой можно узнать, кто работает с файловой системой. Команда `fuser -mv` *`точка_монтирования`* выводит идентификаторы всех процессов, обращающихся к файлам или каталогам указанной файловой системы:

```
$ fuser -mv /usr
            USER     PID      ACCESS   COMMAND
/usr        root     444      ....m    atd
            root     499      ....m    sshd
            root     520      ....m    lpd
. . .
```

Буквенные коды в колонке `ACCESS` указывают на причину, по которой размонтирование не может быть выполнено. Назначение кодов описано в табл. 5.1.

Таблица 5.1. Коды команды `fuser`

Код	Назначение
`f`	Процесс открыл файл для чтения или записи
`c`	В файловой системе находится текущий каталог процесса
`e`	Процесс в данный момент выполняет программу
`r`	В файловой системе находится корневой каталог процесса (задается с помощью команды `chroot`)
`m`	Процесс отображает в памяти файл или совместно используемую библиотеку (обычно неактивный исполняемый файл)

Чтобы в точности определить, что собой представляют эти процессы, вызовите команду `ps`, передав ей список идентификаторов, о которых сообщила команда `fuser`. Например:

```
$ ps -fp "444 499 520"
   UID   PID   PPID  C     STIME    TTY   TIME       CMD
daemon   444   1     0     Apr11    ?     00:00:00   /usr/sbin/atd
  root   499   1     0     Apr11    ?     00:00:23   /usr/sbin/sshd
    lp   520   1     0     Apr11    ?     00:00:00   [lpd]
```

Список идентификаторов взят в кавычки, чтобы интерпретатор передал его команде `ps` как один аргумент.

Команда `fuser` может выдавать статистические сведения об использовании отдельных файлов, а не всей файловой системы. Синтаксис ее вызова в этом случае должен быть таким:`fuser -v` *`имя_файла`*. При наличии опции `-k` команда пошлет всем найденным процессам сигнал `KILL`. Это очень опасное действие, и для его выполнения следует

иметь привилегии пользователя root (им можно стать с помощью утилиты sudo; см. раздел 3.4).

Альтернативой команде fuser служит утилита lsof. Она поддерживается множеством различных версий UNIX и Linux, что делает ее особенно удобной для вызова из сценариев, которые должны работать в различных системах. lsof — более сложная программа, чем fuser, и, соответственно, результаты ее работы более содержательны.

Сценарии, предназначенные для поиска конкретной информации, также располагают возможностью непосредственного чтения файлов, хранящихся в каталоге /proc. Однако команда lsof -F, форматирующая вывод команды lsof для облегчения синтаксического анализа, — более простое и переносимое решение. Для запроса только той информации, которая действительно необходима, следует использовать дополнительные флаги командной строки.

5.3. Организация файловой системы

Файловые системы в UNIX никогда не были хорошо организованы. Одновременно используется много разных, несовместимых соглашений об именовании файлов. Во многих случаях файлы группируются по выполняемым функциям, независимо от того, как часто они изменяются. Это затрудняет обновление операционной системы. Например, каталог /etc содержит ряд файлов, которые никогда не меняются, а также полностью локальные файлы. Как определить, какие файлы нельзя трогать при обновлении системы? Это нужно просто знать...

Такие нововведения, как, например, каталог /var, помогли решить некоторые проблемы, но файлы большинства систем все еще недостаточно упорядочены. Впрочем, для всего находится место. Особенно важно не менять стандартную структуру файлового дерева Linux, поскольку программные пакеты и их инсталляционные утилиты (а также многие системные администраторы!) часто ищут те или иные файлы в строго определенных каталогах.

О конфигурировании ядра рассказывается в главе 28.

Корневая файловая система содержит корневой каталог и минимальный набор файлов и подкаталогов. Файл ядра находится в подкаталоге /boot и обычно его имя начинается с vmlinuz[5]. Частью корневой файловой системы являются также каталог /dev для файлов устройств (за исключением каталога /dev/pts, монтируемого отдельно), каталог /etc для важных системных файлов, каталоги /sbin и /bin для основных утилит и иногда каталог /tmp для временных файлов.

В разделе 7.4 приводятся аргументы в пользу того, зачем разбивать диск на разделы, и рассказывается, как именно это следует делать.

Огромное значение имеют также каталоги /usr и /var. В первом хранится большинство стандартных программ и другие полезные компоненты, в частности интерактивная документация и библиотеки. Совсем не обязательно, чтобы каталог /usr был отдельной файловой системой, однако для удобства администрирования его, как правило, монти-

[5] Когда-то было принято, чтобы каталог /boot представлял собой отдельную файловую систему — в основном, это диктовалось необходимостью размещения ядра вблизи начала загрузочного диска, чтобы оно было доступно для BIOS. Современные ПК не сталкиваются с подобной проблемой, и чаще всего каталог /boot является частью корневой файловой системы.

руют именно так. Чтобы система могла загрузиться в многопользовательском режиме, необходимы оба каталога — /usr и /var.

В каталоге /var содержатся буферные каталоги, журнальные файлы, учетная информация и прочие компоненты, специфичные для каждого компьютера. Поскольку при возникновении проблем журнальные файлы быстро разрастаются, рекомендуется помещать каталог /var в отдельную файловую систему.

Домашние каталоги пользователей чаще всего хранятся в отдельной файловой системе, которая обычно монтируется в корневом каталоге. Отдельные файловые системы можно использовать и для хранения больших информационных массивов, например библиотек исходных кодов программ и баз данных.

Наиболее важные стандартные каталоги перечислены в табл. 5.2 .

В развивающемся стандарте Filesystem Hierarchy Standard (www.pathname.com/fhs) предпринимается попытка систематизировать и унифицировать стандартные каталоги, а также пояснить их назначение. Это прекрасный информационный ресурс, с которым можно сверяться при возникновении вопросов по поводу того, куда инсталлировать тот или иной компонент. Дополнительные соображения, касающиеся иерархии локальных каталогов, приведены в разделе 11.6.

Таблица 5.2. Стандартные каталоги и их содержимое

Каталог	Содержимое
/bin	Команды для обеспечения минимального уровня работоспособности системы
/boot	Ядро и файлы для его загрузки
/dev	Файлы устройств: дисков, принтеров, псевдо-терминалов и т.д.
/etc	Важные файлы запуска и конфигурации системы
/home	Домашние каталоги пользователей
/lib	Библиотеки и компоненты компилятора языка C
/media	Точки монтирования файловых систем на съемных носителях
/opt	Программные пакеты необязательных приложений (которые пока не находят широкого применения)
/proc	Информация о всех выполняющихся процессах
/root	Домашний каталог суперпользователя (часто просто /)
/sbin	Команды загрузки и восстановления системы
/tmp	Временные файлы, которые могут удаляться при перезагрузке
/usr	Иерархия дополнительных файлов и программ
/usr/bin	Большинство команд и исполняемых файлов
/usr/include	Файлы заголовков, предназначенные для компиляции C-программ
/usr/lib	Библиотеки и вспомогательные файлы для стандартных программ
/usr/local	Локальные программы (программы, создаваемые или устанавливаемые локальным пользователем)
/usr/local/bin	Локальные исполняемые файлы
/usr/local/etc	Локальные системные команды и файлы конфигурации

Окончание табл. 5.2

Каталог	Содержимое
/usr/local/lib	Локальные вспомогательные файлы
/usr/local/sbin	Локальные служебные системные команды
/usr/local/src	Исходные коды программ каталогов /usr/local/*
/usr/man	Страницы интерактивной документации
/usr/sbin	Менее важные команды системного администрирования
/usr/share	Элементы, общие для различных систем (доступен только для чтения)
/usr/share/man	Страницы интерактивной документации
/usr/src	Исходные коды нелокальных программных пакетов (не находит широкого применения)
/var	Системные данные и конфигурационные файлы
/var/adm	Разное: журнальные файлы, записи об инсталляции системы, административные компоненты
/var/log	Системные журнальные файлы
/var/spool	Буферные каталоги для принтеров, электронной почты и т.д.
/var/tmp	Каталог для временного хранения файлов (после перезагрузки файлы не исчезают)

5.4. Типы файлов

В Linux определены семь типов файлов. Даже если разработчики добавляют в файловую систему что-нибудь новое и необычное (например, информацию о процессах в каталог /proc), им приходится маскировать это под файлы стандартных типов.

- обычные файлы;
- каталоги;
- файлы байт-ориентированных (символьных) устройств;
- файлы блочно-ориентированных (блочных) устройств;
- локальные сокеты;
- именованные каналы (реализующие принцип обслуживания FIFO — First in First Out ("первым поступил — первым обслужен"));
- символические ссылки.

Определить тип существующего файла можно с помощью команды ls -ld. Первый символ в строке вывода обозначает тип объекта. В следующем примере выдается информация о каталоге /usr/include:

```
$ ls -ld /usr/include
drwxr-xr-x   27 root      root    4096   Jul 15 20:57   /usr/include
```

Возможные коды перечислены в табл. 5.3.

Таблица 5.3. Кодирование типов файлов в листинге команды `ls`

Тип файла	Символ	Создается командой	Удаляется командой
Обычный файл	-	редакторы, `cp` и др.	`rm`
Каталог	d	`mkdir`	`rmdir`, `rm -r`
Файл байт-ориентированного устройства	c	`mknod`	`rm`
Файл блочно-ориентированного устройства	b	`mknod`	`rm`
Локальный сокет	s	`socket(2)`	`rm`
Именованный канал	p	`mknod`	`rm`
Символическая ссылка	l	`ln -s`	`rm`

Как видно из таблицы, команда `rm` является универсальным средством удаления файлов. Но как удалить файл, имя которого, скажем, `-f`? В большинстве файловых систем это абсолютно корректное имя, но команда `rm -f` не сделает то, что нужно, поскольку выражение `-f` будет воспринято как флаг команды. Выйти из положения можно, либо указав более полное имя (например, `./-f`), либо воспользовавшись специальным аргументом `--`, который сообщит команде `rm` о том, что остальная часть командной строки представляет собой имя файла, а не список аргументов: `rm -- -f`

Похожая проблема возникает с файлами, в именах которых присутствуют управляющие символы, поскольку такие имена трудно или вообще невозможно воспроизвести с помощью клавиатуры. В подобной ситуации нужно воспользоваться метасимволами интерпретатора команд, чтобы задавать не имя целиком, а лишь его шаблон. Указывайте также опцию `-i`, чтобы команда `rm` требовала подтвердить удаление каждого файла. Это позволит не удалить нужные файлы, соответствующие шаблону. Вот как, к примеру, можно удалить файл с оригинальным именем `foo<Ctrl+D>bar`:

```
$ ls
foo?bar   foose   kde-root

$ rm -i foo*
rm: remove `foo\004bar'? y
rm: remove `foose'? n
```

Команда `ls` отображает вместо управляющего символа знак вопроса, что иногда сбивает с толку[6]. Если забыть, что символ `?` тоже является подстановочным знаком интерпретатора, и попытаться выполнить команду `rm foo?bar`, можно удалить более одного файла (правда, не в данном примере). Тем выше значимость флага `-i`.

Если имена файлов совсем уж сложно набирать, прибегните к последнему средству:

```
rm -i *
```

Еще одна возможность удаления файлов с причудливыми именами — использование другого интерфейса файловой системы, такого как режим представления каталогов интерпретатора `emacs` или визуального средства, подобного Nautilus.

[6] Команда `ls -b` отображает специальные символы в виде восьмеричных значений, что может пригодиться, если их нужно отдельно идентифицировать. Так символ `<Control-A>` отображается как `1` (`\001` в восьмеричной системе), `<Control-B>` - как `2` и т.д.

Обычные файлы

Обычный файл — это просто последовательность байтов. В Linux не налагаются ограничения на его структуру. Текстовые документы, файлы данных, программные файлы, библиотеки функций и многое другое — все это хранится в обычных файлах. К их содержимому возможен как последовательный, так и прямой доступ.

Каталоги

Каталог хранит именованные ссылки на другие файлы. Он создается командой `mkdir` и удаляется (при условии, что он пуст) командой `rmdir`. Непустые каталоги можно удалить командой `rm -r`.

Специальные ссылки "`.`" и "`..`" обозначают, соответственно, сам каталог и его родительский каталог. Такие ссылки нельзя удалить. Поскольку корневой каталог находится на вершине иерархии, ссылка "`..`" в нем эквивалентна ссылке "`.`".

Имя файла в действительности хранится в родительском каталоге, а не в самом файле. На файл можно ссылаться из нескольких каталогов одновременно и даже из нескольких элементов одного и того же каталога, причем у всех ссылок могут быть разные имена. Это создает иллюзию того, что файл одновременно присутствует в разных каталогах.

Ссылку невозможно отличить от файла, на который она указывает: в Linux они идентичны. Операционная система подсчитывает количество ссылок на каждый файл и при удалении файла не освобождает блоки данных до тех пор, пока не будет удалена последняя ссылка на него. Ссылки не могут указывать на файл, находящийся в другой файловой системе.

Такие ссылки обычно называют жесткими, чтобы отличать их от символических ("мягких") ссылок, которые описаны ниже. Жесткие ссылки создаются командой `ln` и удаляются командой `rm`.

Синтаксис команды `ln` легко запомнить, поскольку она является "зеркальным отражением" команды `cp`. Команда `cp` *`старый_файл новый_файл`* создает опию файла *`старый_файл`* с именем *`новый_файл`*, а команда `ln` *`новый_файл старый_файл`* преобразует имя *`новый_файл`* в дополнительную ссылку на файл *`старый_файл`*.

Важно понимать, что жесткие ссылки не являются особым типом файлов, просто файловая система позволяет создавать несколько ссылок на один файл. Не только содержимое, но и атрибуты файла, в частности, права доступа и идентификатор владельца, являются общими для всех ссылок.

Файлы байт-ориентированных и блочно-ориентированных устройств

📖 Устройства и драйверы более подробно рассматриваются в главе 28.

Файлы устройств позволяют программам получать доступ к аппаратным средствам и периферийному оборудованию системы. На этапе конфигурирования к ядру подключаются те модули, которые "знают", как взаимодействовать с каждым из имеющихся устройств[7]. За управление конкретным устройством отвечает специальный модуль, называемый драйвером устройства.

[7] Ядро способно динамически загружать модули.

Драйверы устройств образуют стандартный коммуникационный интерфейс, который воспринимается пользователем как совокупность обычных файлов. Получив запрос к файлу байт-ориентированного (символьного) или блочно-ориентированного (блочного) устройства, ядро передает этот запрос соответствующему драйверу. Важно отличать файлы устройств от драйверов этих устройств. Файлы сами по себе не являются драйверами. Их можно рассматривать как шлюзы, через которые драйвер принимает запросы.

Файлы символьных устройств позволяют связанным с ними драйверам выполнять собственную буферизацию ввода-вывода. Файлы блочных устройств обрабатываются драйверами, которые осуществляют ввод-вывод большими порциями (блоками), а буферизацию выполняет ядро. В прошлом некоторые типы аппаратных средств могли быть представлены файлами любого типа, но в современных системах такая конфигурация встречается редко.

Файлы устройств характеризуются двумя номерами: старшим и младшим. Старший номер устройства позволяет ядру определить, к какому драйверу относится файл, а младший номер, как правило, идентифицирует конкретное физическое устройство. Например, старший номер устройства 6 в Linux соответствует драйверу параллельного порта. Таким образом, первый параллельный порт (`/dev/lp0`) будет иметь старший номер 6 и младший номер 0.

Драйверы могут интерпретировать переданные им младшие номера устройств как угодно. Например, драйверы накопителей на магнитных лентах с помощью этого номера определяют, необходимо ли перемотать ленту после закрытия файла устройства.

Файлы устройств создаются командой `mknod` и удаляются командой `rm`. Однако необходимость создания файлов устройств вручную возникает редко. В большинстве дистрибутивов команда `udev` используется для автоматического создания и удаления файлов устройств синхронно с обнаружением оборудования ядром. Команда `udev` поддерживает порядок в каталоге `/dev`, ограничивая число фиктивных файлов устройств и поддерживая соответствие между номерами устройств, присваиваемых файлам, и номерами, которые ожидаются ядром. Устройства и драйверы более подробно рассматриваются в главе 28.

Сценарий `MAKEDEV` служит хорошим дополнением команде `udev` в тех случаях, когда требуется создать файлы устройств вручную. Он кодирует обычные имена и номера устройств различных классов, освобождая от необходимости искать эти значения самостоятельно. Например, команда `MAKEDEV pty` создает файлы устройств для псевдо-терминалов.

Информацию о старших и младших номерах, используемых конкретным драйвером устройства, можно найти в 4 разделе его интерактивной документации (например, `man 4 tty`).

Локальные сокеты

Установленные посредством сокетов соединения позволяют процессам взаимодействовать, не подвергаясь влиянию других процессов. В Linux поддерживается несколько видов сокетов, использование которых, как правило, предполагает наличие сети. Локальные сокеты доступны только на локальном компьютере, и обращение к ним осуществляется через специальные объекты файловой системы, а не через сетевые порты. Иногда такие сокеты называют UNIX-сокетами.

📖 Подробнее о системе Syslog рассказывается в главе 10.

Несмотря на то что другие процессы распознают файлы сокетов как элементы каталога, только процессы, между которыми установлено соответствующее соединение, могут осуществлять над файлом сокета операции чтения и записи. С локальными сокетами работают системы печати, X Window и Syslog.

Локальные сокеты создаются с помощью системного вызова `socket`. Когда с обеих сторон соединение закрыто, сокет можно удалить командой `rm` или с помощью системного вызова `unlink`.

Именованные каналы

Подобно локальным сокетам, именованные каналы обеспечивают взаимодействие двух процессов, выполняемых на одном компьютере. Такие каналы еще называют файлами FIFO (First In, First Out — "первым поступил, первым обслужен"). Они создаются командой `mknod` и удаляются командой `rm`.

Символические ссылки

Символическая, или "мягкая", ссылка позволяет вместо имени файла указывать его псевдоним. Столкнувшись при поиске файла с символической ссылкой, ядро извлекает хранящееся в ней имя. Разница между жесткими и символическими ссылками состоит в том, что жесткая ссылка является прямой, т.е. указывает непосредственно на индексный дескриптор файла, тогда как символическая ссылка указывает на файл по имени. Файл, адресуемый символической ссылкой, и сама ссылка представляют собой разные объекты файловой системы.

Символические ссылки создаются командой `ln -s` и удаляются командой `rm`. Они могут содержать произвольное имя, т.е. разрешается указывать на файлы, хранящиеся в других файловых системах, и даже на несуществующие файлы. Иногда несколько символических ссылок образуют петлю.

Символическая ссылка может хранить как полное, так и сокращенное имя файла. Например, команда

```
# ln -s archived/secure /var/log/secure
```

посредством относительного пути связывает имя `/var/log/secure` с именем `/var/log/archived/secure`. Как видно из следующего результата выполнения команды `ls`, полученная символическая ссылка будет содержать строку "`archived/secure`":

```
$ ls -l /var/log/secure
lrwxrwxrwx 1 root root 18 2005-07-05 12:54 /var/log/secure -> archived/secure[8]
```

Каталог `/var/log` можно переместить куда угодно, но символическая ссылка останется корректной.

Часто ошибочно думают, будто первый аргумент команды `ln -s` должен преобразовываться с учетом текущего каталога. На самом деле он не раскрывается командой `ln`, а записывается в символическую ссылку буквально.

[8] Права доступа к файлу, отображаемые командой `ls` для символической ссылки, `lrwxrwxrwx`, представляют собой мнимые значения. Права на создание, удаление или разрешение ссылки управляются содержащим ее каталогом, в то время как права чтения, записи или выполнения для файла, указываемого ссылкой, предоставляются собственными правами целевого файла. Следовательно, символическая ссылка не нуждается (и не обладает) ни в какой собственной информации о правах доступа.

5.5. Атрибуты файлов

В традиционной модели файловой системы UNIX и Linux каждому файлу соответствует набор из девяти битов режима. Они определяют, какие пользователи имеют право читать файл, записывать в него данные или запускать его на выполнение. Вместе с другими тремя битами, которые в основном влияют на работу исполняемых файлов, этот набор образует код, или режим, доступа к файлу.

Двенадцать битов режима хранятся вместе с четырьмя дополнительными битами, определяющими тип файла. Эти четыре бита устанавливаются при создании файла и не подлежат изменению. Биты режима могут изменяться владельцем файла или суперпользователем с помощью команды `chmod` ("change mode" — изменить режим). Просмотр значений этих битов осуществляется с помощью команды `ls -l` (или `ls -ld` в случае каталога). Пример приведен далее в этой главе.

Биты режима

Девять битов режима определяют, кто и какие операции может выполнять над файлом. Традиционная система UNIX не позволяет устанавливать режим на уровне отдельного пользователя (хотя теперь Linux поддерживает списки управления доступом во всех основных файловых системах). Подробнее это описано в разделе 5.6. Вместо этого три различных набора битов определяют права доступа, предоставляемые владельцу файла, членам группы, которой принадлежит файл, и прочим пользователям. Каждый набор состоит из трех битов: бита чтения, бита записи и бита выполнения.

Код доступа удобно записывать в виде восьмеричного числа, так как каждая цифра в нем представляется тремя битами. Три старших бита (в коде доступа им соответствуют восьмеричные значения 400, 200 и 100) служат для управления доступом к файлу его владельца. Вторые три бита (40, 20 и 10) задают доступ для членов группы. Последние три бита (4, 2 и 1) определяют доступ к файлу остальных пользователей. Старший бит каждой триады — это бит чтения, средний — бит записи, младший — бит выполнения.

Каждый пользователь попадает только в одну из категорий прав доступа. В случае неоднозначности выбираются самые строгие права. Например, права владельца файла всегда определяются триадой битов владельца и никогда — битами группы. Возможна ситуация, когда другие пользователи имеют больше прав, чем владелец, но такая конфигурация используется редко.

Для обычного файла бит чтения означает возможность открывать и читать файл. Бит записи позволяет изменять содержимое файла, в том числе полностью очищать его. Правом удаления и переименования файла управляют биты, заданные для его родительского каталога, поскольку именно там хранится имя файла.

Установкой бита выполнения задается разрешение запускать файл. Существует два типа исполняемых файлов: бинарные файлы, которые выполняются непосредственно центральным процессором, и сценарии, обрабатываемые интерпретатором команд или какой-нибудь другой аналогичной программой. В соответствии с принятыми соглашениями сценарии начинаются со строки примерно такого вида:

```
#!/usr/bin/perl
```

Она задает соответствующий интерпретатор. Текстовые исполняемые файлы, в которых не указан конкретный интерпретатор, считаются сценариями `bash` или `sh`[9].

Будучи установленным для каталога, бит выполнения (в этом контексте его часто называют битом поиска) разрешает переходить в каталог или проходить через него при доступе к файлу, но нельзя получить список содержимого каталога. Последнее становится возможным, если для каталога задана комбинация битов чтения и выполнения. Комбинация битов записи и выполнения позволяет создавать, удалять и переименовывать файлы в данном каталоге.

Биты setuid и setgid

Биты, которым в коде доступа соответствуют восьмеричные значения 4000 и 2000, — это биты смены идентификатора пользователя (setuid) и смены идентификатора группы (setgid). Будучи установленным для исполняемых файлов, они позволяют программам получать доступ к файлам и процессам, которые при прочих обстоятельствах недоступны пользователю, выполняющему эти программы. Подробнее этот механизм был описан в разделе 3.1.

Если бит setgid установлен для каталога, то создаваемые в нем файлы будут принимать идентификатор группы каталога, а не группы, в которую входит владелец файла. Это упрощает пользователям, принадлежащим к одной группе, совместный доступ к каталогам. Следует также учитывать, что для исполняемого файла бит setgid имеет другое значение.

Бит setgid можно устанавливать для файлов, не являющихся исполняемыми. Тем самым запрашивается специальный режим блокирования файлов при их открытии. Правда, нам не известны случаи использования этой возможности.

Sticky-бит

Бит, которому в коде доступа соответствует восьмеричное значение 1000, называется sticky-битом (“sticky” — липкий). В первых UNIX-системах sticky-бит запрещал выгрузку программ из памяти. Сегодня он утратил свое значение и попросту игнорируется в современных системах.

Если sticky-бит установлен для каталога, то файловая система позволит удалять и переименовывать его файлы только владельцу каталога, владельцу файла или пользователю `root`. Иметь одно лишь право записи в каталог недостаточно. Такая мера позволяет несколько лучше защитить каталоги вроде `/tmp`.

Просмотр атрибутов файла

Файловая система хранит для каждого файла около сорока информационных полей; большая часть из них используется самой файловой системой. Администратора в основном интересует количество жестких ссылок, владелец, группа, код доступа, время последнего обращения и последней модификации, размер и тип файла. Всю эту информацию можно получить с помощью команды `ls -l` (или команды `ls -ld` в случае каталога).

[9] Когда файл начинается с символов `#!`, он в первую очередь обрабатывается ядром. Но если интерпретатор команд не указан или указан неправильно, ядро откажется выполнять файл. В этом случае текущий интерпретатор предпримет вторую попытку, попытавшись выполнить сценарий в интерпретаторе `sh`.

Для каждого файла хранится также время последнего изменения атрибутов. Традиционное название этого поля ("ctime" от "change time" — время изменения) многих вводит в заблуждение, так как они полагают, что это время создания файла. На самом деле здесь зафиксировано время последнего изменения одного из атрибутов файла (владелец, код доступа и т.д.), но не его содержимого.

Рассмотрим пример:

```
$ ls -l /bin/gzip
-rwxr-xr-x     3 root      root    57136 Jun 15   2004 /bin/gzip
```

В первом поле задается тип файла и режим доступа к нему. Поскольку первый символ — дефис, значит, перед нами обычный файл (см. табл. 5.3).

Следующие девять символов — это три набора битов режима. Порядок наборов таков: владелец, группа, другие пользователи. В листинге команды ls биты режима представляются буквами `r`, `w` и `x` (чтение, запись и выполнение). В показанном примере владелец имеет все права доступа к файлу, а остальные пользователи — только право на чтение и выполнение.

Если бы был установлен бит setuid, то вместо буквы `x`, обозначающей право владельца на выполнение, стояла бы буква `s`. В случае бита setgid вместо буквы `x` для группы тоже была бы указана буква `s`. Последний бит режима (право выполнения для других пользователей) представляется буквой `t`, когда для файла задан sticky-бит. Если бит setuid/setgid или sticky-бит установлен, а надлежащий бит выполнения — нет, эти биты представляются символами `S` и `T`, что указывает на игнорирование данных атрибутов вследствие ошибки.

Второе поле листинга представляет собой счетчик ссылок на файл. В данном случае здесь стоит цифра `3`, свидетельствующая о том, что `/bin/gzip` — одно из трех имен файла (два других — `/bin/gunzip` и `/bin/zcat`). Каждый раз при создании жесткой ссылки на файл этот счетчик увеличивается на единицу.

Любой каталог имеет минимум две жесткие ссылки: одну из родительского каталога и одну из специального файла "`.`" внутри самого каталога. Символические ссылки в счетчике не учитываются.

Следующие два поля определяют владельца и группу файла. В данном примере файл принадлежит пользователю root и одноименной группе. В действительности ядро хранит эти данные не в строковом виде, а в виде идентификаторов. Если символьные версии идентификаторов определить невозможно, в этих полях будут отображаться числа. Так происходит, когда запись пользователя или группы была удалена из файла, соответственно, `/etc/passwd` или `/etc/group`. Не исключено также, что возникла ошибка в базе данных NIS или LDAP (описана в главе 17), если она используется.

В следующем поле отображается размер файла в байтах. Рассматриваемый файл имеет размер 57136 байт, т.е. чуть более 56 Кбайт[10]. Далее указана дата последней модификации файла: 15 июня 2004 г. В последнем поле листинга приведено имя файла: `/bin/gzip`.

Для файла устройства команда `ls` выдает несколько иную информацию. Например:

```
$ ls -l /dev/tty0
crw-rw----   1 root root      4, 0 Jun 11 20:41 /dev/tty0
```

[10] "`K`" обозначает "кило" — метрическую приставку, которой соответствует множитель 1000. В вычислительной технике ей придан несколько иной смысл: 2^{10}, или 1024. Аналогичным образом мегабайт — это не миллион байтов, а 2^{20}, или 1 048 576, байт. Международная комиссия по электротехнике пытается предложить новый набор метрических приставок (таких как киби- и меби-), которые явно обозначают степени числа 2, но как-то не верится, что миллионы людей согласятся поменять привычные обозначения.

Большинство полей осталось тем же, но вместо размера в байтах показаны старший и младший номера устройства. Имя `/dev/tty0` относится к первой виртуальной консоли, управляемой драйвером устройства 4 (драйвер терминала).

При поиске жестких ссылок может пригодиться команда `ls -i`, отображающая для каждого файла номер его индексного дескриптора. Не вдаваясь в детали реализации файловой системы, скажем, что этот номер представляет собой индекс таблицы, в которой перечислены все файлы системы. Именно на индексные дескрипторы ссылаются файловые записи каталогов. Жесткие ссылки, указывающие на один и тот же файл, будут иметь одинаковый номер. Чтобы составить представление о "паутине" ссылок, воспользуйтесь командой `ls -li` для определения числа ссылок и номера индексного дескриптора какого-нибудь файла, а затем поищите его "двойников" с помощью команды `find`[11].

Система автоматически отслеживает время модификации, количество ссылок и размер файла. Права доступа и идентификаторы владельцев файла изменяются явным образом.

Другими важными опциями команды `ls` являются `-a`, отображающая все записи в каталоге (даже те файлы, имена которых начинаются с точки), `-t`, выполняющую сортировку файлов по времени изменения (или `-tr`, осуществляющая сортировку в обратном хронологическом порядке), `-F`, отображающая имена файлов с выделением каталогов и исполняемых файлов, `-R`, выводящая рекурсивный список, и `-h`, которая отображает размеры файлов в удобной для человеческого восприятия форме (например, `8K` или `53M`).

Команда chmod: изменение прав доступа

Код доступа к файлу можно изменить с помощью команды `chmod`. Такое право есть лишь у владельца файла и пользователя root. В первых UNIX-системах код доступа задавался в виде восьмеричного числа. В современных версиях поддерживается также система мнемонических обозначений. Первый способ удобнее для системного администратора, но при этом можно задать только абсолютное значение кода доступа. В случае использования мнемонического синтаксиса разрешается сбрасывать и устанавливать отдельные биты режима.

Первым аргументом команды `chmod` является спецификация прав доступа. Второй и последующий аргументы — это имена файлов, права доступа к которым подлежат изменению. При использовании восьмеричной формы записи первая цифра относится к владельцу, вторая — к группе, а третья — к другим пользователям. Если необходимо задать биты setuid/setgid или sticky-бит, следует указывать не три, а четыре восьмеричные цифры: первая цифра в этом случае будет соответствовать трем специальным битам.

В табл. 5.4 показано восемь возможных комбинаций для каждого трехбитового набора, где символы `r`, `w` и `x` обозначают, соответственно, право чтения, записи и выполнения.

Таблица 5.4. Коды доступа в команде chmod

Восьмеричное число	Двоичное число	Режим доступа	Восьмеричное число	Двоичное число	Режим доступа
0	000	---	4	100	r--
1	001	--x	5	101	r-x
2	010	-w-	6	110	rw-
3	011	-wx	7	111	rwx

[11] Попробуйте команду `find` *точка_монтирования* `-xdev -inum` *индексный_дескриптор* `-print`.

Например, команда `chmod 711 myprog` предоставляет владельцу все права, а остальным пользователям — только право выполнения[12].

Полное описание мнемонического синтаксиса можно найти на man-странице команды `chmod`. В табл. 5.5 приведено несколько примеров мнемонических спецификаций.

Таблица 5.5. Примеры мнемонических спецификаций команды `chmod`

Спецификация	Назначение
`u+w`	Владельцу дополнительно дается право выполнения
`ug=rw,o=r`	Владельцу и членам группы дается право чтения/записи, остальным пользователям — право чтения
`a-x`	У всех пользователей отбирается право выполнения
`ug=srx,o=`	Владельцу и членам группы дается право чтения/выполнения, устанавливаются также биты setuid и setgid; остальным пользователям запрещается доступ к файлу
`g=u`	Членам группы предоставляются такие же права, что и владельцу

Символ `u` (“user”) обозначает владельца файла, `g` (“group”) — группу, `o` (“others”) — других пользователей, `a` (“all”) — всех пользователей.

Права доступа можно “заимствовать” у существующего файла. Например:

```
chmod --reference=filea fileb
```

назначает файлу `filea` такой же код доступа, как и у файла `fileb`.

При наличии опции `-R` команда `chmod` будет рекурсивно обновлять права доступа ко всем файлам указанного каталога и его подкаталогов. Здесь удобнее всего придерживаться мнемонического синтаксиса, чтобы менялись только те биты, которые заданы явно. Например:

```
$ chmod -R g+w mydir
```

добавляет групповое право записи к каталогу `mydir` и его содержимому, не затрагивая остальные права.

Команда `chown`: смена владельца и группы

Команда `chown` предназначена для изменения владельца файла и группы, которой он принадлежит. Первый аргумент команды задается в формате *`пользователь`*`:`*`группа`*. Любой из этих двух атрибутов можно опустить. Если группа не указана, символ двоеточия также не обязателен, но его можно применять, чтобы команда `chown` установила в качестве группы ту, к которой данный *`пользователь`* принадлежит по умолчанию. В целях совместимости с предшествующими версиями файловых систем допускается также использование формы *`пользователь`*`.`*`группа`*, хотя это и не желательно, поскольку имена пользователей могут содержать символы точек.

Для того чтобы изменить группу файла, необходимо либо быть владельцем файла и входить в назначаемую группу, либо быть пользователем root. Поменять владельца файла может только суперпользователь.

[12] Если файл `myprog` является сценарием интерпретатора команд, должно быть задано также право чтения для остальных пользователей. Файлы сценариев, запускаемые интерпретатором, открываются и читаются в текстовом виде, а бинарные файлы выполняются непосредственно ядром, поэтому для них не нужно задавать право чтения.

Как и команда chmod, команда chown имеет флаг -R, который задает смену владельца или группы не только самого каталога, но и всех его подкаталогов и файлов. Например, последовательность команд:

```
# chmod 755 ~matt
# chown -R matt:staff ~matt
```

можно применять для настройки домашнего каталога нового пользователя после копирования в него стандартных конфигурационных сценариев. Не следует пытаться выполнять команду chown для файлов, имена которых начинаются с точки:

```
# chown -R matt:staff ~matt/.*
```

Указанному шаблону соответствует также файл ~matt/.., поэтому будет изменен владелец родительского каталога и, возможно, домашних каталогов других пользователей.

В традиционных UNIX-системах существует отдельная команда chgrp, меняющая группу файла. В Linux тоже есть эта команда, но она полностью дублирует команду chown.

Команда umask: задание стандартных прав доступа

Встроенная команда umask позволяет менять стандартный режим доступа к создаваемым файлам. Значение umask задается в виде трехзначного восьмеричного числа, которое соответствует аннулируемым правам доступа. При создании файла код доступа к нему устанавливается равным разнице между величиной, которую запрашивает создающая файл программа, и значением umask. В табл. 5.6 перечислены возможные комбинации битов режима для команды umask.

Например, команда umask 027 предоставляет все права владельцу файла, запрещает читать файл членам группы и не дает никаких прав другим пользователям. Стандартное значение umask равно, как правило, 022, т.е. право модификации файла предоставляется только его владельцу.

Таблица 5.6. Схема кодирования значения umask

Восьмеричное число	Двоичное число	Режим доступа	Восьмеричное число	Двоичное число	Режим доступа
0	000	rwx	4	100	-wx
1	001	rw-	5	101	-w-
2	010	r-x	6	110	--x
3	011	r--	7	111	---

О стандартных конфигурационных сценариях речь идет в главе 6.

Нельзя заставить пользователей придерживаться конкретного значения umask, так как они в любой момент могут изменить его. Можно, однако, задать стандартное значение umask в тех копиях файлов .profile и .cshrc, которые предоставляются новым пользователям системы.

Дополнительные флаги

В файловых системах ext2fs и ext3fs определен ряд вспомогательных флагов, устанавливая которые, можно запрашивать особые режимы обработки файлов (здесь важно подчеркнуть слово "запрашивать", поскольку многие флаги еще не реализованы). Например, один флаг делает файл доступным только для дополнения, а другой — недоступным для изменения и удаления.

В связи с тем что эти флаги являются специфической особенностью файловых систем серии ext*, в Linux для их просмотра и изменения предназначены специальные команды `lsattr` и `chattr`. В табл. 5.7 перечислены действительно реализованные флаги (в настоящее время только около половины этих флагов описаны в интерактивной документации).

Таблица 5.7. Дополнительные флаги файловых систем ext2fs и ext3fs

Флаг	Назначение
`A`	Никогда не обновлять время доступа (в целях повышения производительности)
`a`	Разрешить запись в файл только в режиме добавления (может устанавливаться лишь суперпользователем)
`D`	Включить режим синхронной записи изменений каталога
`d`	Не создавать резервные копии (команда dump будет игнорировать такой файл)
`i`	Сделать файл неизменяемым и неудаляемым (может устанавливаться лишь суперпользователем)
`j`	Вести журнал изменения данных и метаданных
`S`	Включить режим синхронной записи изменений (буферизация отсутствует)

Не вполне ясно, имеют ли остальные флаги, за исключением `d`, практическую ценность. Флаги `a` и `i` изначально задумывались как дополнительное средство защиты системы от хакеров и злонамеренных программ. К сожалению, они лишь мешают работе полезных программ, отпугивая только тех хакеров, которые не знают о существовании команды `chattr -ia`[13]. Исследования показали, что эти флаги чаще используются самими хакерами, чем для защиты от них.

Особого внимания заслуживают флаги `S` и `D`. Поскольку при внесении изменений они заставляет немедленно записывать на диск все страницы памяти, связанные с файлом или каталогом, может показаться, будто это служит дополнительной защитой данных на случай сбоя. На самом же деле порядок выполнения операций, связанных с синхронными обновлениями, весьма необычен и, как стало известно, мешает работе команды `fsck`. В результате процедура восстановления поврежденной файловой системы усложняется, вместо того чтобы стать более надежной. Обычно лучше использовать возможность ведения журнала файловой системы, поддерживаемую ext3fs. Опция `j` может вызвать запись в журнал информации об определенных файлах (правда, за счет некоторого снижения производительности системы).

[13] Существует специальный механизм (описанный в разделе 20.6), затрудняющий отключение этих флагов, но в настоящее время он не документирован и почти не применяется.

5.6. Списки контроля доступа

9-битовая система контроля доступа (access control system, ACL) "владелец/группа/другие" позволяет решать большинство задач администрирования. Хотя ей присущи явные ограничения, она хорошо согласуется с традициями (кое-кто может сказать "с прошлыми традициями") простоты и предсказуемости UNIX.

Практически все другие операционные системы используют значительно более сложный метод управления доступом к файлам: списки управления доступом или ACL. Списки ACL не имеют фиксированной длины и могут содержать определения прав множества пользователей или групп. Более сложные системы позволяют администраторам указывать частичные наборы прав или отрицательные права, а некоторые из систем поддерживают наследование, которое позволяет определять права доступа на основании нескольких ACL. Конечно, эти системы предоставляют больше возможностей, чем традиционная модель UNIX, но они и сложнее ее на порядок как для администраторов, так и для разработчиков программ.

🕮 Команда `mount` и монтирование файловой системы рассматриваются в разделе 7.4.

В результате усилий по включению ACL в спецификацию POSIX многие варианты UNIX поддерживают в значительной степени стандартизованный механизм ACL, действующий совместно с традиционной 9-битовой моделью UNIX. В Linux поддержка ACL реализована для файловых систем ext2, ext3, ReiserFS, XFS и JFS. Обычно по умолчанию списки управления доступом отключены, и чтобы их смонтировать, необходимо использовать опцию `-o acl`.

Мы описываем модель ACL ради полноты картины. Но не поддавайтесь "чарам" внешней привлекательности — ACL не всегда служат лучшей заменой традиционных прав доступа к файлам, и опытные администраторы прибегают к ним с некоторой опаской. Они не только сложнее в использовании, но и могут порождать проблемы при использовании в сочетании с NFS, системами резервного копирования и такими программами, как текстовые редакторы. Списки ACL тяготеют к беспорядку и поэтому со временем они становятся непригодными для управления.

🕮 Информацию о пакете Samba можно найти в разделе 26.6.

Вероятно, наиболее веская причина использования ACL — стремление повысить совместимость с другими операционными системами. В частности, пакет Samba, позволяющий использовать файлы как в Linux-системах, так и в системах Windows, поддерживает ACL и достаточно успешно выполняет их преобразование между форматами Linux и Windows.

Обзор ACL

Списки ACL Linux представляют собой практически простое расширение стандартной 9-битовой модели. Система поддерживает только права чтения, записи и выполнения. Такие дополнительные атрибуты, как setuid- и sticky-биты, обрабатываются исключительно традиционными битами определения режима.

Списки ACL позволяют устанавливать биты `rwx` независимо для любой комбинации пользователей и групп. Возможный вид отдельных записей в ACL приведен в табл. 5.8.

Таблица 5.8. Записи, которые могут встретиться в списке управления доступом.

Формат	Пример	Определяет права доступа для
`user::`*`права_доступа`*	`user::rw-`	владельца файла
`user:`*`имя_пользователя`*`:`*`права_доступа`*	`user:trent:rw-`	конкретного пользователя
`group::`*`права_доступа`*	`group::r-x`	группы, которая владеет файлом
`group:`*`имя_группы`*`:`*`права_доступа`*	`group:staff:rw-`	конкретной группы
`other::`*`права_доступа`*	`other::---`	всех других пользователей
`mask::`*`права_доступа`*	`mask::rwx`	для всех, кроме владельца и других пользователей[a]

[a] Иногда маски не совсем понятны и они пояснены далее в этом разделе.

Пользователи и группы можно указывать именами или UID/GID. Точное число записей, которые может содержать список ACL, зависит от реализации файловой системы и колеблется от 25 в XFS до практически неограниченного значения в файловых системах ReiserFS и JFS. Файловые системы ext2 и ext3 допускают использование до 32 записей, что, вероятно вполне оправдано с точки зрения обеспечения возможности управления списком.

Команда `getfacl` отображает текущий список ACL файла, а `setfacl` — изменяет или создает его. Команда `setfacl -b` *`файл`* служит для очистки списка ACL, `setfacl -m` *`aclspec файл`* изменяет или дополняет список, а `setfacl -x` *`aclspec файл`* удаляет указанные записи из списка. (При использовании опции `-x` часть указания прав доступа аргумента *`aclspec`* необходимо опустить.) Аргумент *`aclspec`* может содержать несколько записей списка, разделяемых запятыми.

При использовании ACL файлы сохраняют свои исходные биты режима, но при этом автоматически поддерживается целостность атрибутов, и два набора прав доступа не могут вступить в конфликт. Ниже приведен пример автоматического обновления записей ACL в ответ на изменения, выполненные командой `chmod`.

```
$ touch /tmp/example
$ ls -l /tmp/example
-rw-rw-r--  1 garth garth  0 Jun 14 15:57 /tmp/example
$ getfacl /tmp/example
getfacl: Removing leading '/' from absolute path names
# file: tmp/example
# owner: garth
# group: garth
user::rwgroup::
rwother::
r--
$ chmod 640 /tmp/example
$ getfacl --omit-header /tmp/example
user::rwgroup::
r--
other::---
```

Эта принудительная целостность атрибутов позволяет более старым программам, которые даже не подозревают о существовании ACL, достаточно успешно работать в использующей эти списки среде. Однако существует и определенная проблема. Несмотря

на то что ACL-запись `group::` в приведенном примере вроде и отслеживает средний набор традиционных битов режима, это не всегда так.

Чтобы понять, в чем здесь дело, предположите, что унаследованная программа очищает биты разрешения записи во всех трех наборах прав традиционного режима (например, `chmod ugo-w` *`file`*). Понятно, что целью выполнения этой команды является запрет выполнения записи в файле кем-либо. Но что произойдет, если результирующий список ACL выглядит следующим образом?

```
user::r--
group::r--
group:staff:rw-
other::r--
```

С точки зрения унаследованной программы файл выглядит неизменяемым, хотя в действительности он доступен для записи для любого члена группы `staff`. Подобная ситуация нежелательна. Для снижения вероятности недоразумений в Linux приняты следующие правила.

- По определению записи `user::` и `other::` списка ACL идентичны битам режима "владелец" и "другие" традиционного режима файла. Изменение режима ведет к изменению соответствующих записей ACL и наоборот.
- Во всех случаях эффективными правами доступа, предоставляемыми владельцу файла и пользователям, которые не указаны как-либо иначе, являются те, которые указаны в соответствующих записях `user::` и `other::` списка ACL.
- Если файл не имеет явно определенного списка ACL или имеет ACL, который состоит только из одной записи `user::`, одной записи `group::` и одной записи `other::`, эти записи идентичны трем наборам традиционных битов режима. Именно такой случай продемонстрирован в вышеприведенном примере применения команды `getfacl`. (Подобный ACL называют минимальным, и он не требует действительной реализации в виде логически отдельного ACL.)
- В более сложных списках ACL традиционные биты режима соответствуют специальной записи ACL, называемой "маской", а не записи `group::`. Маска ограничивает доступ так, что ACL может предоставлять права *всем* названным пользователям, *всем* названным группам *и* группе, заданной по умолчанию.

Иначе говоря, маска определяет верхний предел прав доступа, которые ACL может присваивать отдельным группам и пользователям. Концептуально эта маска аналогична команде `umask`, за исключением того, что маска ACL действует всегда и указывает предоставленные, а не отобранные права. Записи ACL для названных пользователей, названных групп и группы, заданной по умолчанию, могут содержать права доступа, отсутствующие в маске, но ядро будет просто их игнорировать.

В результате традиционные биты режима не в состоянии уменьшить права, глобально предоставленные списком ACL. Более того, удаление бита из части группы традиционного режима ведет к удалению соответствующего бита в маске ACL и, следовательно, к лишению этих прав всех пользователей, кроме владельца файла и тех, кто относится к категории "другие".

При расширении списка ACL приведенного в предыдущем примере для включения в него записей конкретного пользователя и группы команда `setfacl` автоматически определяет соответствующую маску:

```
$ setfacl -m user::r,user:trent:rw,group:admin:rw /tmp/example
$ ls -l /tmp/example
-r--rw----+  1 garth staff   0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
user::r--
user:trent:rw-
group::r--
group:admin:rw-
mask::rw-
other::---
```

Как видите, команда `setfacl` генерирует маску, которая позволяет вступить в действие всем правам, предоставленным в списке ACL. Для определения маски вручную ее нужно включить в список записей ACL, переданный команде `setfacl`, либо использовать опцию `-n`, чтобы помешать ее повторному генерированию командой `setfacl`.

При попытке получения доступа эффективный UID сравнивается с UID владельца файла. Если они совпадают, права доступа определяются правами `user::` в списке ACL. В противном случае, при наличии соответствия с записью конкретного пользователя в ACL, права определяются этой записью и маской ACL. При отсутствии подходящей записи конкретного пользователя файловая система пытается найти подходящую запись группы, предоставляющую запрошенные права. Эти записи также обрабатываются в сочетании с маской ACL. При отсутствии подходящей записи права доступа определяются записью `other::`.

Следует иметь в виду, что применение традиционной команды `chmod` для манипулирования правами доступа группы в файле, связанном с ACL, сказывается только на маске. Продолжим рассмотрение предыдущего примера:

```
$ chmod 770 /tmp/example
$ ls -l /tmp/example
-rwxrwx---+  1 garth staff   0 Jun 14 15:57 /tmp/example
$ getfacl --omit-header /tmp/example
user::rwx
user:trent:rw-
group::r--
group:admin:rw-
mask::rwx
other::---
```

В данном случае результат выполнения команды `ls` вводит в заблуждение. Несмотря на обширные права, предоставленные группе, ни один из ее членов в действительности не обладает правами выполнения файла на одном лишь основании принадлежности к группе. Чтобы предоставить это право, придется отредактировать список ACL вручную.

Записи, определенные по умолчанию

Кроме типов записей, перечисленных в табл. 5.8, списки ACL каталогов могут содержать записи, определенные “по умолчанию”, которые передаются спискам ACL новых файлов и подкаталогов внутри них. Подкаталоги получают свои записи как в форме записей активного ACL, так и в форме записей, заданных по умолчанию. Следовательно, исходные заданные по умолчанию записи могут со временем распространяться по нескольким уровням иерархии каталогов.

Связь между родительским и дочерними ACL не сохраняется при копировании заданных по умолчанию записей. Изменения в заданных по умолчанию записях родительского каталога не отражается в списках ACL существующих подкаталогов.

5.7. Упражнения

5.1. Что такое значение `umask`? Задайте такое значение `umask`, которое запрещало бы членам группы и рядовым пользователям доступ к создаваемым файлам.

5.2. В чем разница между жесткими и символическими ссылками? Когда предпочтительнее использовать ссылки того или иного типа?

☆ 5.3. Прочитайте man-страницу файла `/etc/fstab` и добавьте в него запись, позволяющую автоматически монтировать NTFS-раздел на этапе начальной загрузки. Воспользуйтесь точкой монтирования `/mnt/win_c`.

☆ 5.4. При инсталляции Linux важно разбить жесткий диск на разделы таким образом, чтобы для каждой файловой системы (`/var`, `/usr` и т.д.) было достаточно свободного места. Предположим, в дистрибутиве "Foobar Linux" используются следующие установки:

`/`	100 Мбайт
`/var`	50 Мбайт
`/boot`	10 Мбайт
<раздел подкачки>	128 Мбайт
`/usr`	оставшееся свободное место

Какие проблемы при такой конфигурации могут возникнуть на интенсивно эксплуатируемом сервере?

☆ 5.5. Почему рекомендуется помещать некоторые разделы (в частности, `/var`, `/home` и раздел подкачки) на отдельные диски, чтобы отделить их от файлов данных и программ? А как насчет каталога `/tmp`? Приведите аргументы по каждой из перечисленных файловых систем.

☆ 5.6. Напишите сценарий, который находит все жесткие ссылки в системе.

☆ 5.7. Какими командами реализуются перечисленные ниже действия?

а) Предоставьте владельцу файла `README` право чтения/записи, а остальным пользователям — только право чтения.

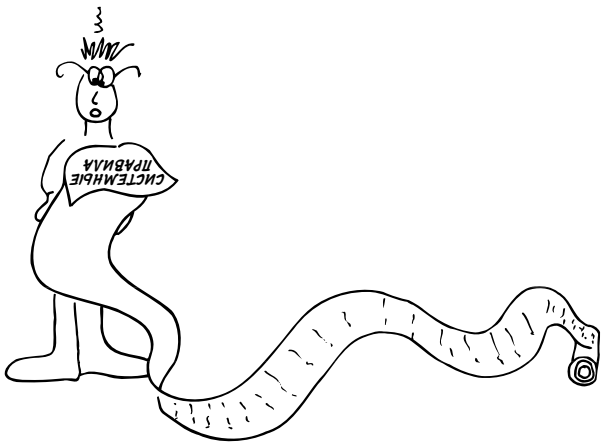
б) Установите для файла бит setuid, не меняя текущие права доступа (и даже не просматривая их).

в) Отобразите содержимое текущего каталога, отсортировав список по дате последней модификации, причем файлы, которые редактировались последними, должны стоять в конце списка.

г) Поменяйте группу файла `shared` с “user” на “friends”.

Глава 6

Подключение новых пользователей



Подключение новых пользователей и удаление старых — рутинная задача в большинстве систем. Выполнение этих задач не представляет особой сложности, но достаточно трудоемко. Большинство администраторов создает различные средства автоматизации этого процесса, а затем поручают выполнение реальной работы своему заместителю или оператору.

В наши дни мы являемся свидетелями возвращения эпохи централизованных серверов с сотнями учетных записей, которые теперь используются наряду с распределенными серверами, предназначенными для обслуживания всего нескольких пользователей. Администраторы должны глубоко понимать подсистему учета пользователей, чтобы уметь настраивать работу сетевых служб и надлежащим образом конфигурировать локальные учетные записи.

Правильное управление учетными записями является залогом безопасности системы. Редко используемые учетные записи становятся главными мишенями атак злоумышленников, как и те записи, пароли к которым легко подобрать. Даже если для подключения и удаления пользователей применяются автоматизированные системные утилиты, важно понимать, какие при этом происходят изменения в системе.

В этой главе мы сначала рассмотрим базовую модель управления учетными записями, реализуемую автоматизированными утилитами, а затем познакомимся с самими утилитами (`useradd`, `userdel` и др.).Стандартная команда `useradd` работает вполне хорошо и удовлетворяет потребностям большинства организаций. К сожалению, этого нельзя сказать о команде `userdel`.

6.1. Файл /etc/passwd

Файл /etc/passwd содержит список пользователей, которые известны системе. В процессе регистрации пользователя система обращается к этому файлу в поисках идентификатора пользователя и его домашнего каталога. Каждая строка файла описывает одного пользователя и содержит семь полей, разделенных двоеточиями:

- регистрационное имя;
- шифрованный пароль или "заполнитель" пароля (вопросы применения шифрованных паролей описаны далее в этом разделе);
- идентификатор пользователя;
- идентификатор группы по умолчанию;
- поле GECOS (полное имя, номер офиса, рабочий и домашний телефоны);
- домашний каталог;
- регистрационная оболочка.

Вот примеры правильно составленных записей файла /etc/passwd:

```
root:lga5FjuGpZ2so:0:0:The System,,x6096,:/:/bin/sh
jl:x:100:0:Jim Lane,ECT8-3,,:/staff/jl:/bin/sh
dotty:$1$Ce8QpAQI$L.DvJEWiHlWetKTMLXFZO/:101:20::/home/dotty:/bin/tcsh
```

В наши дни совершенно недопустимо хранить зашифрованные пароли в текстовом виде. Современные быстрые компьютеры позволяют взламывать такие пароли за считанные минуты. Все версии UNIX и Linux позволяет спрятать их, поместив в отдельный файл, недоступный для чтения. Это называется механизмом скрытых паролей и (по вполне понятным причинам) применяется в большинстве дистрибутивов.

Механизм скрытых паролей становится понятнее, когда его описывают в качестве расширения к традиционному файлу /etc/passwd, поэтому мы отложим его рассмотрение до раздела 6.2. Дополнительная информация о значимости файла скрытых паролей приведена в разделе 20.5.

Файл /etc/passwd часто используется несколькими системами через административную СУБД, такую как NIS или LDAP. Подробнее об этом рассказано в главе 17.

Ниже рассматривается назначение отдельных полей файла /etc/passwd.

Регистрационное имя

📖 Подробнее о СУБД NIS рассказывается разделе 17.4.

Регистрационные имена (называемые также пользовательскими именами) должны быть уникальными и состоять не более чем из 32 символов. Они могут содержать любые символы, кроме двоеточия и символа новой строки. Если используется база данных NIS, длина имени ограничена 8 символами независимо от операционной системы.

В некоторых старых версиях UNIX допустимый диапазон ограничивался алфавитно-цифровыми символами, а длина имени не должна была превышать 8 символов. В смешанных системах рекомендуется придерживаться наиболее строгих ограничений. Это дает возможность избежать конфликтов со старыми программами, а также использовать одно и то же регистрационное имя на любом компьютере. Помните: если сегодня в вашей организации используются однотипные компьютеры, то это не значит, что такое положение вещей сохранится завтра.

Регистрационные имена могут включать как строчные, так и прописные буквы. Однако документ RFC822 призывает не обращать внимания на регистр в адресах электронной почты. Смешение регистров в именах не приводит к возникновению проблем, но имена, состоящие из строчных букв, считаются традиционными, к тому же их проще вводить.

Следует выбирать такие регистрационные имена, которые легко запомнить, поэтому имя, состоящее из случайной последовательности букв, является не самым удачным вариантом. Избегайте также кличек и прозвищ, даже если они допустимы в вашей организации: имена типа DarkLord или Butthead часто включаются в адреса электронной почты, так что не забывайте о самоуважении.

Поскольку регистрационные имена часто используются в адресах электронной почты, полезно определить стандартную процедуру их формирования. Пользователям должна быть предоставлена возможность делать обоснованные предположения о регистрационных именах друг друга. Имена, фамилии, инициалы и сочетания этих элементов — вот приемлемые варианты для схем формирования имен.

🕮 Почтовые псевдонимы подробно рассматриваются в разделе 18.4.

Любая жестко заданная схема в конечном счете приводит к появлению дубликатов или слишком длинных имен, поэтому иногда придется делать исключения. В случае появления длинного имени можно использовать предоставляемую почтовой системой функцию определения псевдонимов, чтобы задать две версии одного и того же имени, по крайней мере, для электронной почты.

Например, схема выбора имен может быть такой: первый инициал и фамилия каждого сотрудника. Пользователь Брент Браунинг (Brent Browning), таким образом, превратится в “bbrowning”, но девять символов — это слишком много для некоторых систем. Лучше присвоить пользователю регистрационное имя “brentb”, а “bbrowning” сделать элементом файла `aliases`:

```
bbrowning: brentb
```

Если в организации имеется глобальный файл почтовых псевдонимов, то все новые регистрационные имена должны отличаться от псевдонимов, присутствующих в этом файле. В противном случае почта будет доставляться не новому пользователю, а пользователю с соответствующим псевдонимом.

В крупных организациях в адресах электронной почты часто используются полные имена (например, `John.Q.Public@mysite.com`), благодаря чему регистрационные имена скрываются от внешнего мира. Это прекрасная идея, но, чтобы облегчить работу администраторам, необходимо установить четкое и понятное соответствие между регистрационными именами и реальными именами пользователей.

🕮 Вопросы эквивалентности регистрационных имен рассматриваются в разделе 20.9.

Регистрационные имена должны быть уникальными в двух отношениях. Во-первых, необходимо, чтобы имя пользователя на всех компьютерах было одинаковым. Это удобно как самому пользователю, так и администратору.

Во-вторых, регистрационное имя всегда должно относиться к одному и тому же лицу. Некоторые команды (например, `ssh`) можно настроить так, чтобы они аутентифицировали удаленных пользователей по регистрационным именам. Даже если адреса `scott@boulder` и `scott@refuge` относятся к разным пользователям, то в случае неправильной настройки учетных записей оба пользователя могут при определенных обстоятельствах получить доступ к файлам друг друга, не вводя пароль.

Опыт также показывает, что дублирующиеся имена сбивают с толку пользователей почтовых систем. Почтовая система различает, кому именно принадлежат имена, однако пользователи часто посылают письма не по тому адресу.

Зашифрованный пароль

Прежде чем переходить к деталям, напомним: в большинстве систем зашифрованные пароли хранятся в файле `/etc/shadow`, а не `/etc/passwd`. Тем не менее, излагаемый материал остается достоверным независимо от местонахождения паролей.

Пароли хранятся в зашифрованном виде. Если только вы не осуществляете шифрование в уме (в таком случае мы очень хотели бы с вами познакомиться), необходимо либо установить содержимое этого поля с помощью команды `passwd` (или `yppasswd`, если используется база данных NIS), либо скопировать строку, содержащую зашифрованный пароль, из другой учетной записи.

Вручную редактируя файл `/etc/passwd` для создания новой учетной записи, поставьте звездочку (`*`) или символ `x` в поле зашифрованного пароля. Звездочка воспрепятствует несанкционированному использованию учетной записи до установки реального пароля. Никогда не оставляйте это поле пустым, иначе безопасность системы подвергнется серьезному нарушению, поскольку для доступа к такой учетной записи пароль не требуется. Даже если используется файл скрытых паролей, не оставляйте файл `/etc/passwd` на растерзание хакерам. Не исключено, что какая-нибудь старая программа обратится к этому файлу в процессе принятия решения, связанного с безопасностью[1].

Основные дистрибутивы Linux поддерживают несколько методов шифрования паролей, и они могут определять, какой из них был использован для шифрования каждого из паролей. Таким образом, не обязательно, чтобы все пароли были зашифрованы по одному алгоритму.

В большинстве дистрибутивов Linux по умолчанию используется алгоритм MD5. С криптографической точки зрения алгоритм MD5 лишь не многим более устойчив, чем DES, зато в нем допускаются пароли произвольной длины. Чем длиннее пароли, тем они надежнее. Поскольку алгоритм MD5 весьма универсален и безопасен, мы рекомендуем применять его во всех системах, которые его поддерживают.

Зашифрованные пароли имеют постоянную длину (34 символа в случае MD5 и 13 символов в случае DES) независимо от длины исходного пароля. Пароли шифруются с добавлением случайной “примеси”, чтобы одному исходному паролю соответствовало много зашифрованных форм. Таким образом, факт выбора пользователями одинаковых паролей не может быть выявлен путем просмотра зашифрованных паролей. Пароли MD5 легко распознать, так как они всегда начинаются с последовательности “1”.

По умолчанию в системе SUSE для шифрования новых паролей применяется алгоритм Blowfish. Как и алгоритм MD5, этот алгоритм обеспечивает надежное шифрование и его выбор по умолчанию вполне оправдан. Однако пароли, зашифрованные этим алгоритмом SUSE, нельзя копировать в другие системы, т.к. только SUSE понимает их. Пароли, зашифрованные алгоритмом Blowfish, легко узнать по префиксу $2a$.

[1] Вот как комментирует эту ситуация Джон Корбет — один из наших технических обозревателей: “Если вы не знаете, в какой момент принимаются решения, связанные с безопасностью — у вас большие проблемы. Подобные вещи не должны заставать администраторов врасплох”.

Идентификатор пользователя

Идентификатор пользователя (UID) — это 32-битное целое число. Но для обеспечения совместимости со старыми системами мы рекомендуем, чтобы значение самого старшего идентификатора по возможности не превышало 32767 (наибольшее 16-битное целое число со знаком).

По определению пользователь root имеет идентификатор 0. В большинстве систем есть также псевдопользователи `bin`, `daemon` и ряд других. Как правило, такие псевдоимена помещаются в начало файла `/etc/passwd`, и им назначаются маленькие идентификаторы. Их не следует присваивать реальным интерпретаторам команд. Чтобы зарезервировать побольше номеров для неперсонифицированных пользователей, мы рекомендуем присваивать реальным пользователям идентификаторы, начиная с номера 500 (или выше).

📖 Об утилите `sudo` речь шла в разделе 3.4.

Нежелательно создавать более одной учетной записи с идентификатором 0. Может показаться удобным иметь несколько суперпользовательских записей с разными интерпретаторами команд и паролями, но в действительности это лишь создает дополнительные бреши в системе защиты и приводит к ненужным трудностям. Если нескольким пользователям необходимо быть администраторами, пусть они применяют утилиту `sudo`.

Избегайте повторного использования идентификаторов, даже идентификаторов тех пользователей, которые уволились из организации и учетные записи которых были удалены. Такая мера предосторожности позволит избежать путаницы, если файлы впоследствии будут восстанавливаться из резервных копий, где пользователи могут быть идентифицированы по номерам, а не по регистрационным именам.

📖 Более подробно NFS рассматривается в главе 16.

Идентификаторы должны быть уникальными в пределах всей организации, т.е. конкретный идентификатор должен соответствовать одному и тому же регистрационному имени и физическому лицу на каждом компьютере. Если уникальность идентификаторов нарушена, то в такой системе, как NFS, появятся проблемы безопасности; кроме того, это может привести в замешательство пользователей, переходящих из одной рабочей группы в другую[2].

Трудно соблюдать уникальность идентификаторов, когда группы компьютеров находятся в административном подчинении разных лиц и даже организаций. Это проблема как техническая, так и концептуальная. Лучшим ее решением является создание центральной базы данных, содержащей для каждого пользователя уникальную запись (мы у себя самостоятельно создали такую базу данных). Проще всего назначить каждой группе в пределах организации отдельный диапазон идентификаторов и позволить распоряжаться им по своему усмотрению. В результате решится вопрос с уникальностью пользовательских идентификаторов (необходимое условие в случае совместного использования файловых систем посредством NFS), но останется проблема уникальности регистрационных имен. Популярным средством управления идентификаторов пользователей становится также база данных LDAP.

[2] Еще одна проблема, связанная с NFS, — идентификатор “бесправного” пользователя, который традиционно применяется для ограничения доступа удаленных суперпользователей. Подробнее этот идентификатор рассматривается в разделе 16.1.

Идентификатор группы по умолчанию

Как и идентификатор пользователя, идентификатор группы (GID) является 32-битным целым числом. Идентификатор 0 зарезервирован для группы с именем root, идентификатор 1 — для группы `bin`, а 2 — для группы `daemon`.

О каталогах с установленным битом setgid рассказывалось в разделе 5.5.

Группы определяются в файле `/etc/group`, а поле идентификатора группы в файле `/etc/passwd` задает стандартный ("эффективный") идентификатор на момент регистрации пользователя в системе. Этот идентификатор не играет особой роли при определении прав доступа[3]; он используется лишь при создании новых файлов и каталогов. Новые файлы обычно включаются в эффективную группу своего владельца, но если у каталога установлен бит setgid (02000) или файловая система смонтирована с опцией `grpid`, новые файлы включаются в группу своего каталога.

Поле GECOS[4]

Поле GECOS в основном используется для хранения персональной информации о каждом пользователе. Оно не имеет четко определенного синтаксиса. Первоначально в Bell Labs его использовали для хранения регистрационной информации, необходимой при передаче пакетных заданий из UNIX-систем мэйнфрейму, работающему под управлением GECOS (General Electric Comprehensive Operating System — комплексная операционная система компании General Electric). Сейчас осталось одно название. Некоторые программы вместо символа "&" в поле GECOS подставляют регистрационное имя пользователя, что облегчает ввод. И `finger`, и `sendmail` выполняют эту подстановку, но многие программы не делают этого. Поэтому на эту функцию лучше не рассчитывать.

В принципе структура поля GECOS может быть произвольной, но команда `finger` интерпретирует разделенные запятыми элементы поля в следующем порядке:

- полное имя (часто используется только это поле);
- номер офиса и здания;
- рабочий телефон;
- домашний телефон.

О базе данных LDAP можно почитать в разделе 17.5.

С помощью команды `chfn` можно изменять информацию, содержащуюся в поле GECOS. Эта команда используется для ведения и обновления списка телефонных номеров, но ею часто злоупотребляют: пользователь может изменить информацию так, что она станет нецензурной или некорректной. Администраторы сетей большинства университетских городков отключают команду `chfn`. Поле GECOS — первоочередной кандидат на управления его содержимым с помощью службы LDAP.

---

[3] При определении прав доступа Linux учитывает членство во всех группах. Версии интерпретаторов команд, предшествующие 2.6.4, допускали принадлежность пользователя к 32 группам, но более поздние версии не накладывают никаких ограничений.

[4] Когда компания Honeywell приобрела компьютерный филиал компании GE, имя поля GECOS было изменено на GCOS. В настоящее время можно встретить оба написания.

Домашний каталог

Войдя в систему, пользователь попадает в свой домашний каталог. Если на момент регистрации этот каталог отсутствует, выводится сообщение вроде "no home directory" (домашний каталог отсутствует)[5]. Если в качестве значения поля домашнего каталога по умолчанию DEFAULT_HOME в файле /etc/login.defs установлено no, продолжение регистрации пользователя будет невозможно; в противном случае пользователь попадает в корневой каталог.

Если домашние каталоги смонтированы через NFS, то в случае проблем с сервером или с самой сетью они могут оказаться недоступными.

Регистрационная оболочка

В качестве регистрационной оболочки, как правило, задается интерпретатор команд, например /bin/sh или /bin/csh, но в принципе это может быть любая программа. По умолчанию используется интерпретатор bash, который запускается, если соответствующее поле в файле /etc/passwd не задано. В Linux имена csh и sh являются всего лишь ссылками на bash (GNU-версия sh) и tcsh (надмножество команд csh), соответственно. Во многих дистрибутивах есть также открытая версия интерпретатора ksh.

Пользователи могут менять интерпретатор с помощью команды chsh. Файл /etc/shells содержит список тех интерпретаторов, которые доступны для выбора. В SUSE этот список является обязательным для соблюдения, а в Red Hat будет лишь выдано предупреждение в случае выбора незарегистрированного интерпретатора. Проверьте, содержат ли элементы файла /etc/shells полные имена команд.

6.2. Файл /etc/shadow

Файл /etc/shadow доступен для чтения только суперпользователю и предназначен для хранения зашифрованных паролей подальше от любопытных глаз. В нем также содержится учетная информация, которая отсутствует в файле /etc/passwd. В одних дистрибутивах механизм скрытых паролей используется по умолчанию, в других — конфигурируется в виде дополнительного пакета.

При использовании скрытых паролей соответствующие поля в файле /etc/passwd всегда содержат символ x.

Файл shadow не включает в себя файл passwd, и последний не генерируется автоматически при изменении файла shadow. Оба файла необходимо сопровождать независимо друг от друга (или использовать команду наподобие useradd для автоматического управления файлами). Как и /etc/passwd, файл /etc/shadow содержит одну строку для каждого пользователя. Каждая строка состоит из 9 полей, разделенных двоеточиями:

- регистрационное имя;
- зашифрованный пароль;
- дата последнего изменения пароля;
- минимальное число дней между изменениями пароля;

[5] Это сообщение появляется при регистрации в системе через консоль или терминал, но не через экранный диспетчер, такой как xdm, gdm или kdm. В последнем случае пользователь вообще будет немедленно выведен из системы, поскольку программа-диспетчер не сможет осуществить запись в нужный каталог (например, ~/.gnome).

- максимальное число дней между изменениями пароля;
- количество дней до истечения срока действия пароля, когда выдается предупреждение;
- количество дней по истечении срока действия пароля, когда учетная запись аннулируется;
- срок действия учетной записи;
- зарезервированное поле, которое в настоящее время всегда пустое.

📖 Подробнее о программе `usermod` рассказывается в разделе 6.7.

Лишь первые два поля обязательно должны быть заполнены. Поля дат задаются в виде числа дней (а *не* секунд), прошедших с 1-го января 1970 г. Это отличается от стандартного способа вычисления времени в UNIX. К счастью, задавать поля можно с помощью команды `usermod`.

Типичная запись выглядит так:

```
millert:$1$buJ6v3Ch$BwLIoF5eaCh9Nv.OEzD3T0:13348:0:180:14::14974:
```

Вот более подробное описание каждого поля.

- Регистрационное имя берется из файла `/etc/passwd`. Оно связывает записи файлов `passwd` и `shadow`.
- Зашифрованный пароль идентичен тому, который ранее хранился в файле `/etc/passwd`.
- Третье поле содержит дату последнего изменения пользователем своего пароля. Это поле обычно заполняется программой `/usr/bin/passwd`.
- В четвертом поле задано количество дней, спустя которые пользователь сможет снова изменить пароль. Его назначение — принудительное изменение регистрационной информации путем препятствования немедленному возврату пользователей к привычным паролям после их обязательного изменения. Эта возможность кажется нам опасной, если в течение указанного времени будет обнаружено нарушение безопасности системы. Поэтому мы рекомендуем устанавливать данное поле равным нулю.
- В пятом поле задано максимальное число дней между двумя изменениями пароля. С помощью этой установки администраторы заставляют пользователей менять свои пароли (подробнее механизм устаревания паролей описан в разделе 20.5). Максимальное время жизни пароля определяется суммой значений данного и седьмого полей.
- В шестом поле задано количество дней, оставшихся до момента устаревания пароля, когда программа `login` должна предупреждать пользователя о необходимости сменить пароль.
- Седьмое поле определяет, спустя какое число дней после устаревания пароля отменяется учетная запись. Его назначение не совсем понятно.
- В восьмом поле задана дата отмены учетной записи. По прошествии этой даты пользователь не сможет зарегистрироваться в системе, пока администратор не сбросит значение поля. Если поле оставлено пустым, учетная запись всегда будет активной.
- Девятое поле зарезервировано на будущее.

Теперь, когда назначение полей стало понятным, вернемся к рассмотренному выше примеру:

```
millert:$1$buJ6v3Ch$BwLIoF5eaCh9Nv.OEzD3T0:13348:0:180:14:: 14974:
```

В этой записи говорится о том, что пользователь `millert` последний раз менял свой пароль 18 июля 2006 г. Следующий раз пароль должен быть изменен через 180 дней. За две недели до этого пользователь начнет получать предупреждения о необходимости сменить пароль. Учетная запись действительна до 31 декабря 2010 г.

С помощью утилиты `pwconv` можно согласовать содержимое файлов `/etc/shadow` и `/etc/passwd`. Большинство параметров файла `shadow` берется из стандартных установок файла `/etc/login.defs`.

Следующий пример иллюстрирует формат файла `login.defs`. Комментарии облегчают понимание назначения различных параметров. Данный конкретный пример взят из системы Fedora. Как содержимое этого файла, определяемое по умолчанию, так и параметры, которые могут быть указаны, в значительной мере зависят от конкретного дистрибутива.

```
# *ОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ*
#  Каталог размещения почтовых ящиков, _или_ имя файла, относительно
#  домашнего каталога. В случае _определения_ обоих параметров MAIL_DIR
#  имеет приоритет.
#  QMAIL_DIR - для Qmail

#  QMAIL_DIR        Maildir
MAIL_DIR            /var/spool/mail
#  MAIL_FILE        .mail

#  Параметры устаревания паролей:
#  PASS_MAX_DAYS    Максимальное число дней возможного использования пароля.
#  PASS_MIN_DAYS    Минимальное число дней между изменениями паролей.
#  PASS_MIN_LEN     Минимальная допустимая длина пароля.
#  PASS_WARN_AGE    Число дней до истечения срока действия пароля,
                    когда выдается предупреждающее сообщение.

PASS_MAX_DAYS       99999
PASS_MIN_DAYS       0
PASS_MIN_LEN        5
PASS_WARN_AGE       7

#  Минимальное/максимальное значения для автоматического выбора
#  идентификаторов пользователей в программе useradd

UID_MIN             500
UID_MAX             60000

#  Минимальное/максимальное значения для автоматического выбора
#  идентификаторов групп в программе groupadd

GID_MIN             500
GID_MAX             60000

#  Если она определена, эта команда запускается при удалении пользователя.
#  Она должна удалять любые процессы в каталоге/cron/print, принадлежащие
```

```
# удаляемому пользователю (имя которого передано в качестве первого аргумента).

# USERDEL_CMD      /usr/sbin/userdel_local

# Если по умолчанию команда useradd должна создавать домашние каталоги
# пользователей, в системах RH systems выполняется это действие. Этой
# опцией является ORed с установленным флагом -m в командной строке useradd.

CREATE_HOME        yes
```

6.3. Файл /etc/group

Файл `/etc/group` содержит имена UNIX-групп и списки членов каждой группы. Например:

```
wheel:x:10:trent,ned,evi,garth,lynda,boggs,millert
csstaff:*:100:lloyd,evi
student:*:200:dotty
```

Запись файла представляет одну группу и содержит четыре поля:

- имя группы;
- зашифрованный пароль или символ `x`, указывающий на использование файла `gshadow`;
- идентификатор группы;
- список членов, разделенный запятыми (пробелов быть не должно).

Как и в файле `/etc/passwd`, поля разделяются двоеточиями. Длина имени группы не должна превышать 8 символов из соображений совместимости, хотя в Linux такого ограничения нет. Несмотря на наличие поля пароля (с помощью которого пользователи могут присоединиться к группе, выполнив команду `newgrp`) пароль задается редко[6]. В большинстве случаев в это поле вводятся звездочки, но можно оставить его пустым. Команда `newgrp` не сменит группу без пароля, если только пользователь не указан как член этой группы. Все дистрибутивы, приведенные в качестве примеров, распространяются сконфигурированными для использования файлов `/etc/gshadow`, которые по своей сути аналогичны файлу `/etc/shadow`, но имеют значительно меньшее значение (поскольку пароли групп используются редко).

Имена групп и их идентификаторы должны быть одинаковыми на всех компьютерах, получающих совместный доступ к файлам через NFS. Этого трудно достичь в гетерогенной среде, поскольку в разных операционных системах используются разные идентификаторы для одних и тех же групп. Мы считаем, что по умолчанию пользователь не должен регистрироваться как член системной группы.

Если в файле `/etc/passwd` пользователь объявлен членом определенной группы, а в файле `/etc/group` — нет, пользователь все равно включается в группу. На этапе регистрации членство в группе проверяется по обоим файлам, но лучше все же согласовывать их содержимое.

Чтобы избежать конфликтов, связанных с идентификаторами стандартных групп, рекомендуем выбирать идентификаторы локальных групп, начиная со значения 500.

[6] Единственная известная нам причина, по которой пользователю Linux может понадобиться команда `newgrp`, заключается в назначении стандартной группы новым файлам.

Согласно традиции, новые пользователи добавляются в группу, название которой отражает их категорию, например “students” или “finance”. Однако следует учитывать, что подобная традиция повышает вероятность доступа пользователей к файлам друг друга из-за неаккуратной установки прав. Чтобы этого избежать, мы рекомендуем создавать для каждого пользователя уникальную группу. Лучше всего выбирать одинаковое имя для пользователя и группы. Можно также выбрать одинаковый номер пользователя и группы

В такую персональную группу должен входить лишь сам пользователь. Если нужно позволить другим пользователям совместно работать с файлами посредством группового владения, создайте для этих целей отдельную группу. Идея персональных групп состоит не в том, чтобы запретить механизм группового доступа как таковой, а в том, чтобы создать более строгую стандартную группу для каждого пользователя и предотвратить случайный коллективный доступ к файлам.

Утилита `useradd` всех выбранных в качестве примеров дистрибутивов, за исключением SUSE, по умолчанию помещает пользователей в персональные группы.

6.4. Подключение пользователей

Прежде чем создавать учетную запись для нового пользователя в правительственном учреждении, учебном заведении или в крупной коммерческой компании, крайне важно заставить его подписать соглашение о правилах работы в системе. (Как?! У вас нет такого соглашения? Немедленно прочтите материал в разделе 30.11, чтобы узнать, для чего оно нужно и как его составлять.)

У пользователей нет веских причин подписывать такое соглашение, поэтому в интересах администратора убедить их сделать это, чтобы иметь рычаги влияния. После того как учетная запись создана, добиться подписи будет трудно, так что лучше получить ее заранее.

Процесс подключения нового пользователя состоит из ряда этапов. Четыре из них определяются системными требованиями, два связаны с формированием пользовательской среды, а еще несколько этапов может понадобиться для целей системного администрирования. Обязательные этапы:

- отредактировать файлы `passwd` и `shadow` с целью создания учетной записи пользователя;
- добавить запись нового пользователя в файл `/etc/group`;
- задать пароль;
- создать для нового пользователя домашний каталог, назначив его владельца с помощью команды `chown` и задав режим доступа с помощью команды `chmod`.
- Пользовательские этапы:
- скопировать в домашний каталог пользователя стандартные конфигурационные сценарии;
- создать каталог электронной почты и настроить почтовые псевдонимы.
- Административные этапы:
- проверить правильность создания учетной записи;
- добавить контактную информацию о пользователе, а также сведения о статусе учетной записи в административную базу данных.

Описанная в разделе команда useradd и ее аналоги автоматически выполняют некоторые из перечисленных операций, но в следующих подразделах мы опишем все действия так, как если бы их пришлось осуществлять вручную. Это необходимо для понимания того, что делают автоматизированные утилиты. На практике предпочтительнее (быстрее и в меньшей степени чревато ошибками) использовать команду useradd или похожий конфигурационный сценарий.

Все описываемые операции необходимо выполнять с правами суперпользователя или воспользоваться утилитой sudo. Об утилите sudo рассказывалось в разделе 3.4.

Редактирование файлов /etc/passwd и /etc/shadow

Чтобы безопасно отредактировать файл passwd, выполните команду vipw, которая запустит текстовый редактор с копией файла. По умолчанию выбирается редактор vi, но эту установку можно изменить, задав новое значение переменной среды EDITOR. Существование временной копии файла служит своего рода блокировкой: команда vipw позволяет только одному пользователю редактировать файл passwd. Когда пользователь выходит из редактора, команда vipw заменяет исходный файл passwd отредактированной копией.

В системах Fedora и RHEL по завершении редактирования файла passwd команда vipw автоматически запрашивает о необходимости редактирования файла shadow. В системах SUSE, Debian и Ubuntu для этого необходимо использовать команду vipw -s.

Например, для создания учетной записи tyler необходимо добавить в файл /etc/passwd следующую строку:

```
tyler:x:2422:2422:Tyler Stevens, ECEE 3-27, x7919,:/home/tyler:/bin/sh
```

Нужно также добавить соответствующую запись в файл /etc/shadow:

```
tyler:*::::::14974:
```

В этой строке говорится о том, что у пользователя tyler нет зашифрованного пароля, механизм устаревания пароля не включен, а учетная запись действительна до 31 декабря 2010 г.

Редактирование файла /etc/group

Затем в файле /etc/group потребуется создать запись персональной группы пользователя tyler, которой мы также присвоим имя tyler. Этой группе нужно присвоить номер 2422, поскольку именно эту группу мы назначили данному пользователю по умолчанию в файле passwd[7].

```
tyler::2422:tyler
```

Строго говоря, пользователь tyler будет в группе номер 2422 независимо от того, указан ли он в файле /etc/group, потому что это членство уже предоставлено ему благодаря записи файла /etc/passwd. Содержимое файлов /etc/passwd и /etc/group не имеет значения для ядра. Оно оперирует только идентификаторами пользователя и группы. Основная цель записи информации о персональных группах в файл group — обеспечение правильности отображения имен этих групп такими командами, как ls.

[7] Такое “именование” и “нумерация” — всего лишь дань традиции; подробнее эти вопросы были описаны в разделе 6.3.

Безусловно, всегда лучше иметь под рукой достоверный список созданных групп и входящих в них пользователей.

Если бы пользователя `tyler` нужно было включить в дополнительные группы, достаточно было бы включить его регистрационное имя в эти группы в файле `/etc/group`.

Задание пароля

Суперпользователь может изменить пароль любого пользователя с помощью следующей команды:

```
# passwd user
```

или

```
$ sudo passwd user
```

📖 Правила выбора паролей приведены в разделе 20.5.

Команда `passwd` запросит новый пароль и потребует повторить его. Если введен короткий пароль, состоящий только из строчных букв, или какая-нибудь другая строка, неприемлемая в качестве пароля, команда `passwd` потребует задать что-нибудь более сложное. В большинстве систем Linux предлагаемые пароли сверяются со словарем, чтобы пользователи не вводили пароли, которые можно легко подобрать.

Утилита `mkpasswd`, входящая в пакет `expect`, позволяет генерировать случайные пароли для новых пользователей. Хорошо это или плохо, но назначение случайного пароля заставляет пользователя немедленно сменить его, потому что такой пароль труден для запоминания[8]. Не путайте утилиту `mkpasswd` со стандартной командой `mkpasswd`, которая просто кодирует заданную строку в формате пароля.

Никогда не оставляйте без пароля новую учетную запись или запись с доступом к интерпретатору команд.

Создание домашнего каталога пользователя

Каждый каталог, создаваемый суперпользователем, изначально принадлежит пользователю root, поэтому необходимо изменить его владельца и группу с помощью команды `chown`. Следующая группа команд создает домашний каталог для пользователя `tyler`:

```
# mkdir /home/tyler
# chown tyler:staff /home/tyler
# chmod 700 /home/tyler
```

Копирование стандартных конфигурационных файлов

Работу некоторых команд и утилит можно настроить, поместив их конфигурационные файлы в свой домашний каталог. Имена таких файлов традиционно начинаются с точки и заканчиваются суффиксом `rc` (сокращение от “run command” — команда запуска). Предшествующая точка заставляет команду `ls` не включать эти файлы в листинги

[8] В действительности пароли являются не случайными, а псевдослучайными. Теоретически, если один или более паролей, сгенерированных псевдослучайным образом, взломан, обратный анализ последовательности символов позволяет определить дополнительные пароли. Такой сценарий возможен, но маловероятен в реальной жизни. Поэтому риск применения этого подхода сравнительно невелик.

каталогов, если только не указана опция -a. Наиболее часто встречающиеся конфигурационные файлы перечислены в табл. 6.1.

Таблица 6.1. Распространенные конфигурационные файлы

Утилита	Имя файла	Применение
csh/tcsh	.login	Установка типа терминала (если это необходимо)
		Установка опций программ biff и mesg
	.cshrc	Установка переменных среды
		Установка псевдонимов команд
		Установка переменной среды PATH
		Установка значения umask
		Установка переменной cdpath
		Установка строки приглашения, формирование списка выполненных команд
bash[a]	.bashrc	Аналог файла .cshrc для интерпретатора bash
	.bash_profile	Аналог файла .login для интерпретатора bash
vim	.vimrc	Установка опций редактора vim
emacs	.emacs	Установка опций редактора emacs
		Функциональная привязка клавиш редактора emacs
mail/mailx	.mailrc	Задание персональных почтовых псевдонимов
		Установка параметров почтового клиента
xrdb[б]	.Xdefaults	Задание параметров конфигурации X11: шрифты, цвета и т.д.
startx[б]	.xinitrc	Задание начальной среды X11
	.Xclients	Задание начальной среды X11 (RHEL, Fedora)
xdm[б]	.xsession	Задание начальной среды X11

[a] Интерпретатор bash читает также файл .profile или /etc/profile для эмуляции интерпретатора sh.

[б] Конкретные особенности X Windows зависят от реализации и используемого диспетчера окон. Подробнее эти вопросы рассмотрены в главе 22.

Если у вас еще нет набора универсальных конфигурационных файлов, создайте их в каталоге /usr/local/lib/skel. Скопируйте какие-нибудь похожие файлы и отредактируйте их с помощью текстового редактора. Лучше всего воспользоваться заготовками, оставленными разработчиками системы в каталоге /etc/skel. Не забудьте задать разумное значение umask (мы рекомендуем 077, 027 или 022, в зависимости от "дружелюбности" среды и размеров организации).

В зависимости от интерпретатора в каталоге /etc могут содержаться системные конфигурационные файлы, обрабатываемые раньше пользовательских.

Например, интерпретатор bash читает файл /etc/profile до начала обработки файла ~/.bash_profile. Информацию о других интерпретаторах можно найти в документации.

Как правило, интерпретаторы ищут дополнительные системные конфигурационные файлы в каталоге /etc/profile.d. Это соглашение позволяет программным пакетам единообразно указывать используемые по умолчанию конфигурационные параметры уровня интерпретатора. Например, в системах Fedora и RHEL файлы /etc/profile.d/colorls.* содержат параметры профиля technicolor для вывода команды ls.

Последовательность команд инсталляции конфигурационных файлов для нового пользователя tyler такова:

```
# cp /usr/local/lib/skel/.[a-zA-Z]* ~tyler
# chown tyler:staff ~tyler/.[a-zA-Z]*
# chmod 600 ~tyler/.[a-zA-Z]*
```

Нельзя использовать команду

```
# chown tyler:staff ~tyler/.*
```

потому что пользователь tyler станет владельцем не только собственных файлов, но и родительского каталога ".." (/home). Это очень распространенная и опасная ошибка системного администратора.

Назначение каталога для электронной почты

Пользователи предпочитают получать электронную почту на каком-то одном компьютере. Эта возможность часто реализуется путем добавления записи в файл глобальных псевдонимов /etc/mail/aliases или установки параметра userDB программы sendmail на центральном почтовом сервере. Электронная почта рассматривается в главе 18, а способы организации почтовых каталогов — в разделе 18.3.

Проверка новой учетной записи

Чтобы проверить, правильно ли сформирована новая учетная запись, выйдите из системы, а затем зарегистрируйтесь как новый пользователь и выполните следующие команды:

```
$ pwd           /* проверка домашнего каталога */
$ ls -la        /* проверка принадлежности конфигурационных файлов */
```

Администратор должен сообщить новым пользователям их регистрационные имена и исходные пароли. Во многих организациях эта информация пересылается по электронной почте, но так поступать нежелательно из соображений безопасности: учетная запись нового пользователя может оказаться взломанной еще до того, как он успеет зарегистрироваться в системе.

📖 Подробнее о письменных соглашениях с пользователями рассказывается в разделе 30.11.

Если в организации принято, чтобы пользователи подписывали соглашение о правилах работы в системе, то не забудьте выполнить эту процедуру до создания учетной записи. Это предотвратит возможные недоразумения и укрепит правовую базу санкций, которые, не исключено, впоследствии придется применять администратору. Кроме того, не забудьте напомнить новым пользователям о необходимости немедленной замены паролей. Это требование можно ужесточить, установив короткий срок устаревания паролей[9].

[9] Поскольку один и тот же пароль может иметь множество шифрованных представлений, этот метод обеспечивает лишь переустановку пароля пользователем, но не выбор действительного нового пароля. Единственно действенным способом принуждения пользователей к действительной замене паролей является ведение базы данных всех ранее использованных значений.

Запись статуса учетной записи и контактной информации о пользователе

Когда администратор лично знаком с каждым пользователем, несложно проследить, кто работает в системе и что делает. Но если в подчинении администратора большой и часто меняющийся штат пользователей, необходим более формальный способ контроля учетных записей. Ведение базы данных с контактной информацией и сведениями о статусе учетных записей поможет, в случае необходимости, легко выяснить, кто есть кто, и зачем тому или иному пользователю понадобилась учетная запись. Контактную информацию лучше держать под рукой, чтобы можно было быстро связаться с нужным пользователем при возникновении проблем.

6.5. Удаление пользователей

Когда пользователь покидает организацию, его учетная запись и файлы должны быть удалены из системы. Эта процедура включает удаление всех ссылок на регистрационное имя, которые были введены вручную или с помощью команды `useradd`. При ручном удалении учетной записи последовательность операций будет примерно такой:

- удалить записи пользователя из локальных баз данных и телефонных списков;
- удалить пользовательские псевдонимы из файла `aliases` либо организовать перенаправление поступающих ему сообщений;
- удалить пользовательские задания из `crontab`-файла, а также из очереди команды `at`;
- уничтожить пользовательские процессы, которые еще выполняются;
- удалить записи пользователя из файлов `passwd`, `shadow`, `group` и `gshadow`;
- удалить домашний каталог пользователя;
- удалить почтовый каталог пользователя.

Перед тем как удалять домашний каталог пользователя, необходимо переместить из него в другие каталоги все файлы, которые нужны остальным пользователям. Поскольку не всегда можно с уверенностью сказать, какие файлы понадобятся, а какие — нет, лучше предварительно скопировать пользовательские домашний и почтовый каталоги на какой-то носитель.

После удаления учетной записи пользователя убедитесь, что в системе не осталось файлов с его идентификатором. Чтобы узнать точный путь к этим файлам, воспользуйтесь командой `find` с аргументом `-nouser`. Поскольку команда `find` может вести поиск на сетевых серверах, проверяйте файловые системы по отдельности, задавая опцию `-xdev`:

```
# find filesystem -xdev -nouser
```

Если в организации за пользователями закреплены отдельные рабочие станции, эффективнее всего переинсталлировать систему, прежде чем предоставлять ее новому пользователю. Но не забудьте сбросить на системный жесткий диск локальные файлы, которые могут понадобиться в будущем.

6.6. Отключение учетной записи

Иногда нужно временно отключить учетную запись пользователя. Проще всего сделать это, поставив звездочку (*) или какой-то другой символ в поле зашифрованного пароля в файле `/etc/shadow`. Эта мера препятствует большинству типов доступа, управляемых паролем, поскольку дешифровка больше не может привести к получению какого-либо приемлемого пароля. Однако такие команды как `ssh`, которые не всегда проверяют системные пароли, могут продолжать работать.

Во всех дистрибутивах, рассматриваемых нами в качестве примеров, за исключением SUSE, команды `usermod -L` *`пользователь`* и `usermod -U` *`пользователь`* позволяют легко блокировать и разблокировать пароли.

Другой (и, по всей видимости, более безопасный) способ получения аналогичного конечного результата — замена интерпретатора команд пользователя программой, которая выдает сообщение, поясняющее, почему учетная запись отключена, и содержащее инструкции по исправлению ситуации. Этот псевдоинтерпретатор не должен быть указан в файле `/etc/shells`. Многие демоны, предоставляющие нерегистрационный доступ к системе (например, `ftpd`), проверяют, упомянут ли интерпретатор пользователя в файле `/etc/shells`. Если он там не указан, вход в систему будет запрещен (именно это нам и требуется). К сожалению, поясняющее сообщение нельзя увидеть, если пользователь пытается зарегистрироваться в системе в оконной среде.

Правда, имеется одна проблема. По умолчанию программа `sendmail` не доставляет почту тем пользователям, чьи интерпретаторы не указаны в файле `/etc/shells`. Как правило, не рекомендуется вмешиваться в доставку почты, даже если получатель не может прочитать ее немедленно. Чтобы “обмануть” программу `sendmail`, добавьте в файл `/etc/shells` ложный интерпретатор с именем `/SENDMAIL/ANY/SHELL` (конечно, это может вызвать нежелательные побочные эффекты).

6.7. Управление учетными записями

Команда `useradd` добавляет записи о пользователях в файл `/etc/passwd` (и в файл `/etc/shadow`, если он есть). Она имеет интерфейс командной строки и легко запускается вручную или из локального сценария `adduser`. Команда `usermod` изменяет записи файла `/etc/passwd` для существующих пользователей. Команда `userdel` удаляет учетную запись из системы, при необходимости уничтожая и домашний каталог пользователя. Команды `groupadd`, `groupmod` и `groupdel` выполняют аналогичные действия по отношению к файлу `/etc/group`.

Вот как можно добавить в систему нового пользователя `hilbert` (со стандартными системными установками):

```
# useradd hilbert
```

Эта команда создает в файле `/etc/passwd` приведенную ниже запись. Обратите внимание на то, что поле пароля содержит звездочку. Таким образом, доступ к учетной записи невозможен до тех пор, пока не будет назначен реальный пароль.

```
hilbert:*:105:20::/home/hilbert:/bin/bash
```

По ряду причин SUSE использует аналогичный, но независимо разработанный макет команд манипулирования пользователями и группами. Эти команды имеют те же имена, но существенно отличаются значением некоторых параметров и поведением, заданным по умолчанию. Например, большинство дистрибутивов создают для новых пользователей персональные группы, если администратор не указывает ничего другого в командной строке. Команда `useradd` дистрибутива SUSE помещает новых пользователей в группу с номером 100. (В конфигурации, определенной по умолчанию, она включает их также в группы "video" и "dialout". Странно, однако...)

Команда `useradd` имеет ряд полезных аргументов. В следующем примере мы указываем, что основной группой пользователя `hilbert` является группа `faculty` и что он также входит в группу `famous`. Помимо этого мы задаем другой домашний каталог и просим команду `useradd` создать его, если он еще не существует.

```
# useradd -c "David Hilbert" -d /home/math/hilbert -g faculty -G famous -m
      -s /bin/sh hilbert
```

В результате в файле `/etc/passwd` появится следующая запись:

```
hilbert:x:1005:30:David Hilbert:/home/math/hilbert:/bin/sh
```

Значение UID здесь на единицу больше, чем значение самого большого из используемых в системе идентификаторов. Соответствующая запись в файле `shadow` будет такой:

```
hilbert:!:11508:0:99999:7:0::
```

Кроме того, пользователь `hilbert` будет добавлен в группы `faculty` и `famous` в файле `/etc/group`. Появится также каталог `/home/math/hilbert`, заполненный на основании содержимого каталога `/etc/skel`.

Во всех выбранных в качестве примеров дистрибутивах, за исключением SUSE, узнать текущие установки можно, выполнив команду `useradd -D`. Флаг `-D` можно использовать в сочетании с другими аргументами для задания стандартных установок.

Даже в системе SUSE начальные параметры хранятся в файле `/etc/default/useradd` и могут быть отредактированы вручную.

Команда `usermod` модифицирует существующую учетную запись, принимая те же опции, что и команда `useradd`. Например, следующая команда задает конечный срок существования учетной записи `hilbert` — 4 июля 2007 г.:

```
# usermod -e 2007-07-04 hilbert
```

Команда `userdel` уничтожает учетную запись, отменяя все изменения, сделанные командой `useradd`. Чтобы удалить пользователя `hilbert` из системы, достаточно ввести

```
# userdel hilbert
```

Эта команда удалит все ссылки на учетную запись `hilbert` из файлов `passwd`, `shadow` и `group`. По умолчанию начальный каталог пользователя не уничтожается[10]. Опция `-r` заставляет команду `userdel` удалить и начальный каталог тоже, но даже в этом, самом "агрессивном" режиме команда выполняет только последние три операции из списка, приведенного в разделе 6.5.

[10] В своей системе мы обычно не удаляем домашние каталоги пользователей в течение нескольких недель. Это позволяет уменьшить число операций восстановления с резервных лент в том случае, если пользователь вернется в систему или другим пользователям понадобится доступ к его рабочим файлам.

Из-за этого в большинстве крупных организаций создаются локальные сценарии `adduser` и `rmuser` (чаще всего они пишутся на языке Perl). Для выполнения части своих задач эти сценарии могут вызывать стандартные утилиты.

6.8. Упражнения

6.1. Как определить стандартную группу, в которую входит пользователь? Как изменить ее?

6.2. Объясните разницу между следующими значениями `umask`: 077, 027, 022, и 755. Как сделать одно из этих значений стандартной общесистемной установкой для новых пользователей? Можно ли заставить пользователей придерживаться одного и того же значения `umask`?

6.3. Каково назначение файла скрытых паролей?

☆ 6.4. Перечислите действия, которые необходимо выполнить для добавления пользователя в систему без помощи команды `useradd`. Какие действия дополнительно необходимы в вашей организации?

☆ 6.5. Разработайте схему выбора пользовательских имен для вашей организации. Какие правила устанавливает данная схема? Как обеспечить уникальность имен? Есть ли недостатки у этой схемы? Как происходит удаление пользователей?

☆☆ 6.6. Возьмите список имен (например, из телефонного справочника) и используйте его в качестве входной информации для сценария, генерирующего регистрационные имена в соответствии с разработанной выше схемой. Сколько учетных записей удалось получить до возникновения конфликта? Сколько вообще конфликтов возникло? На основании полученных результатов дайте оценку схеме выбора пользовательских имен и предложите улучшения к ней.

☆☆ 6.7. Напишите сценарий проверки файла `/etc/passwd`, выполняющий перечисленные ниже действия (для осуществления второго и последнего пунктов может потребовать доступ с правами суперпользователя):

а) поиск записей с идентификатором пользователя, равным нулю;

б) поиск записей без пароля (требуется файл `/etc/shadow`);

в) поиск группы записей с одинаковыми идентификаторами пользователей;

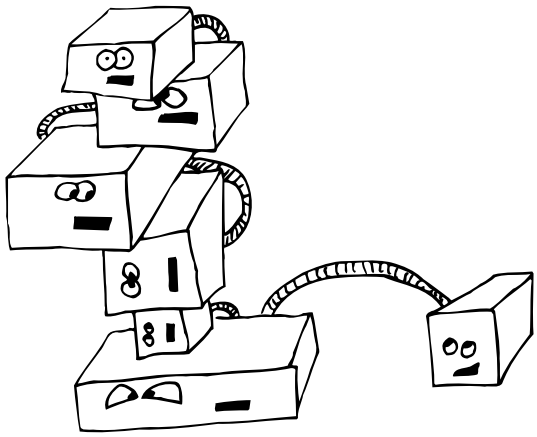
г) поиск записей с одинаковыми регистрационными именами;

д) поиск записей, для которых не установлен срок действия (требуется файл `/etc/shadow`).

Book_LinuxAdmHand.indb 156 17.04.2007 13:54:47

Глава 7

Добавление жесткого диска



В это трудно поверить, но факт остается фактом: несмотря на огромный прогресс систем программного и сетевого обеспечения и несмотря на революционные изменения технологий производства микросхем, способы долговременного хранения данных остались практически теми же, что и 40 лет назад! Конечно, плотность данных возросла (а цены уменьшились) на несколько порядков, и все же базовые концепции остались неизменными.

Вместе с тем люди придумывают все новые способы использования дискового пространства, особенно на фоне повсеместного внедрения Internet-технологий. Во многих ситуациях ограниченность доступного дискового пространства остается главной проблемой. Широкое применение аудиофайлов в формате MP3, потокового видео и другого мультимедийного содержимого постоянно держит системных администраторов в напряжении. Поэтому рациональное управление дисками ничуть не утратило своей актуальности.

Во многих серверных системах дисковые накопители подключаются к стандартной шине периферийных устройств, называемой SCSI (Small Computer Systems Interface — интерфейс малых вычислительных систем). Альтернативный интерфейс IDE (Integrated Drive Electronics — встроенный интерфейс накопителей) является стандартом для персональных и портативных компьютеров. Мы начнем эту главу с общего описания интерфейсов SCSI и IDE и конструкции современных жестких дисков. Далее будут рассмотрены общие механизмы форматирования и разбивки дисков на разделы, а также процедура инициализации файловых систем. И в заключение мы рассмотрим более сложные технологии управления дисковым пространством, такие как массивы RAID и диспетчеры томов.

7.1. Дисковые интерфейсы

В настоящее время широко используется всего несколько стандартных дисковых интерфейсов, хотя на горизонте появился ряд новых технологий. Важно уметь выбрать жесткий диск, соответствующий интерфейсу той системы, в которой он будет установлен. Если система поддерживает несколько интерфейсов, выбор нужного из них должен быть сделан на основании критериев скорости, избыточности, мобильности и цены.

- PATA (называемый также IDE) — простой и дешевый интерфейс для персональных компьютеров. Он был назван "встроенным", потому что предполагал наличие аппаратного контроллера, расположенного непосредственно в корпусе жесткого диска. Взаимодействие процессора и жесткого диска осуществлялось на достаточно высоком уровне. Сегодня это стандартный интерфейс современных жестких дисков. IDE-диски обладают средней и выше среднего скоростью работы, высокой емкостью и невероятно дешевы. Подробнее интерфейс IDE рассмотрен в следующем разделе.
- Интерфейс Serial ATA (последовательный интерфейс ATA) (SATA) — наследник интерфейса IDE. Кроме того, что он поддерживает значительно более высокие скорости передачи данных, интерфейс SATA упрощает подключение дисков, поскольку позволяет применять более тонкие кабели большей длины. Этот интерфейс обеспечивает поддержку "горячей" замены диска и (в ряде случаев) очередизации команд — две функции, которые, наконец, позволили интерфейсу IDE составить серьезную конкуренцию SCSI в серверной среде.
- Хотя в настоящее время интерфейс SCSI не столь распространен, как ранее, тем не менее, он остается одним из наиболее широко поддерживаемых дисковых интерфейсов. Существует несколько его разновидностей, обеспечивающих разную скорость работы. Все они допускают подключение к шине нескольких жестких дисков. Более подробное описание этого интерфейса приведено в разделе "Интерфейс SCSI" далее в этой главе.
- Fibre Channel — последовательный интерфейс, завоевывающий популярность в корпоративной среде благодаря высокой пропускной способности и большому числу устройств, которые можно одновременно подключать к каналу. Устройства соединяются между собой посредством волоконно-оптического или коаксиального медного кабеля. В настоящее время интерфейс обеспечивает скорость передачи данных 100 Мбит/с и выше. Волоконно-оптические сети могут строиться по топологии FC-AL (Fibre Channel Arbitrated Loop — волоконно-оптическая петля с арбитражной логикой) или с помощью коммутируемых фабрик (сверхбыстродействующих коммутаторов). В таких сетях применяются самые разные протоколы, включая IP. Волоконно-оптическое устройство идентифицируется по аппаратному номеру, который называется глобально уникальным именем (World Wide Name) и напоминает MAC-адрес Ethernet.
- USB — ставший популярным в последнее время стандарт подключения таких устройств, как клавиатура и мышь, обладающий при этом достаточной пропускной способностью для обслуживания дисковых накопителей и дисководов CD-ROM. Интерфейс USB широко распространен в персональных компьютерах и позволяет легко перемещать диски между системами.

Интерфейсы SCSI и IDE доминируют на рынке дисковых систем, поэтому их мы и будем подробно рассматривать.

Интерфейс PATA

Интерфейс PATA (Parallel Advanced Technology Attachment — усовершенствованная технология параллельного подключения), также называемый IDE, задумывался как простое и недорогое решение. Чаще всего он применяется в персональных компьютерах и рабочих станциях среднего класса. Интерфейс IDE завоевал популярность в конце 80-х гг. Вскоре после этого появился стандарт ATA-2, удовлетворявший растущие запросы потребителей и изготовителей жестких дисков.

В стандарт ATA-2 были добавлены режимы PIO (Programmed I/O — программируемый ввод-вывод) и DMA (Direct Memory Access — прямой доступ к памяти), а также расширенные средства поддержки технологии Plug-and-Play. Кроме того, был реализован режим LBA (Logical Block Addressing — логическая адресация блоков), который (в сочетании с расширенным кодом BIOS персональных компьютеров) позволял системе получать доступ не только к первым 1024 цилиндрам диска. Ранее из-за подобного ограничения максимальный размер диска составлял 540 Мбайт. Тогда никто и не думал, что когда-нибудь понадобятся диски большей емкости!

Поскольку BIOS управляет процессом начальной загрузки компьютера, раньше приходилось создавать маленький загрузочный раздел в пределах первых 1024 цилиндров, чтобы ядро загружалось старой BIOS-программой. После того как ядро активизировалось и начинало выполняться, необходимость в BIOS отпадала и появлялась возможность доступа к остальной части диска. Подобный обходной маневр не нужен в современных системах, поскольку в режиме LBA вместо схемы адресации "цилиндр-головка-сектор" используется линейная схема.

В стандарт ATA-3 добавлены средства повышения надежности, управления питанием и самодиагностики. Спецификация Ultra-ATA является попыткой устранить разрыв между стандартами ATA-3 и ATA-4 за счет добавления высокопроизводительных режимов Ultra DMA/33 и Ultra DMA/66. В них пропускная способность шины расширяется с 16 Мбайт/с до 33 Мбайт/с и 66 Мбайт/с, соответственно. В стандарте ATA-4 преследовалась цель объединить интерфейсы ATA-3 и ATAPI (ATA Packet Interface — пакетный интерфейс ATA). Последний представляет собой протокол, позволяющий дисководам CD-ROM и накопителям на магнитной ленте подключаться к шине IDE.

Последующие члены семейства, ATA-5 и ATA-6, включают расширенные средства обработки ошибок и управления производительностью, особенно полезные в такой многопользовательской среде, как Linux. Предполагается, что недавно разработанный интерфейс ATA-7 станет последним в ряду параллельных интерфейсов ATA. Он поддерживает скорости обмена данными до 133 Мбит/с.

IDE-диски почти всегда являются внутренними (иногда в процессе тестирования возникает необходимость расположить жесткий диск рядом с системным блоком или на нем; такой диск можно назвать "внешним"). Максимальная длина кабеля для шины ATA-2 составляет всего 18 дюймов (45 см), из-за чего дисковод иногда трудно установить даже в верхний отсек. Помимо жесткого ограничения на длину кабеля существует еще одно неудобство: интерфейс IDE позволяет подключать к шине только два устройства. Для преодоления этого недостатка некоторые производители оснащают материнские платы несколькими шинами IDE (как правило, двумя, называемыми соответственно основной и дополнительной).

Доступ к IDE-устройствам первых модификаций осуществлялся последовательно, т.е. в определенный момент времени активным могло быть только одно устройство. Наивысшая производительность этих устаревших дисководов IDE достигалась при их подключении к нескольким шинам. Стандарт расширенного интерфейса IDE (Extended IDE — EIDE) обходит это ограничение, имитируя наличие двух интерфейсов IDE в одном интерфейсе, что обеспечивает возможность одновременного использования двух активных дисководов. Конечно, устройства не могут одновременно передавать данные по кабелю, но это же ограничение относится и к интерфейсу SCSI.

В IDE используется 40-контактный разъем, расположенный на плоском кабеле, который соединяет диск с интерфейсной платой. В новых IDE-стандартах, в частности в Ultra DMA/66, применяется другой кабель с большим числом линий "земли", вследствие чего уровень шумов понижен. Если кабель или жесткий диск не маркированы должным образом, убедитесь, что первый контакт дискового разъема соединен с первым контактом интерфейсной платы. Контакт номер 1 обычно помечается маленькой цифрой 1 на разъемном соединении. Если этой цифры нет, воспользуйтесь правилом, которое гласит, что первый контакт расположен ближе всего к разъему питания. На кабеле первый контакт обозначается красной полоской. Если такая маркировка крайнего провода кабеля отсутствует, просто совместите первый контакт кабеля с первым контактом разъема интерфейса и пометьте кабель для удобства его использования в будущем.

Если к шине IDE подключено несколько устройств, необходимо одно из них сделать главным, а другие — подчиненными. Установка перемычки на современных дисководах в положение "cable select" (выбор посредством кабеля), обычно выбранное по умолчанию, позволяет устройствам самостоятельно выбирать режим работы в качестве главного или подчиненного. Работа устройства в качестве главного не обеспечивает никаких преимуществ в плане производительности. Некоторые старые IDE-диски не желают быть подчиненными; в таком случае нужно попробовать поменять устройства ролями. Если они по-прежнему не работают, лучше подключить каждое устройство к отдельной шине.

Устанавливая IDE-устройства, необходимо помнить о следующем.

- Новые IDE-диски работают со старыми платами, а старые IDE-диски работают с новыми платами. Естественно, поддерживаются только те возможности, которые являются общими для обоих типов устройств.
- Максимальная длина кабеля равная всего 18 дюймам, слишком мала, поэтому дополнительное устройство может подключаться к шине с натяжкой. Если устройство работает нестабильно, проверьте, не отходит ли где-нибудь разъем.
- Новые технологии изготовления кабелей позволяют использовать для подключения устройств IDE круглые, а не плоские кабели. Использование новых кабелей позволяет сделать их прокладку внутри системного блока более аккуратной, и способствуют улучшению вентиляции.

Интерфейс SATA

С повышением скоростей передачи данных между дисководами PATA (особенно с появлением интерфейса ATA-7), недостатки стандарта стали очевидными. На высоких скоростях передачи данных стали проявляться проблемы надежности, связанные с электромагнитными и другими электрическими помехами. Интерфейс Serial ATA (SATA) разрабатывался для решения этих проблем. Хотя концепция интерфейса SATA была разработана уже к 2000 г., лишь недавно она нашла свое воплощение в пользовательских рабочих станциях.

Интерфейс SATA сглаживает многие "острые углы" интерфейса PATA. Хотя первые дисководы SATA лишь незначительно превосходят по производительности интерфейс ATA-7, который обеспечивает скорость передачи, равную 150 Мбит/с, можно не сомневаться, что к 2008 г. интерфейс SATA-2 будет поддерживать скорость равную 600 Мбит/с[1]. Другими значительными изменениями являются прекрасная технология проверки ошибок, возможность "горячей смены" диска, встроенные очереди команд и ряд других технических усовершенствований. И, наконец, интерфейс SATA устраняет необходимость назначения главных и подчиненных устройств, поскольку только одно устройство может быть подключено к каждому каналу.

Интерфейс SATA снимает ограничение PATA на максимальную длину кабеля, равную 18 дюймам, и вводит новые стандарты кабелей и соединителей соответственно с 7 и 15 контактами[2]. Эти кабели значительно более гибкие и с ними значительно легче работать, чем с их плоскими предшественниками — больше не нужно изгибать и скручивать кабели, чтобы подключить дисководы к одному и тому же кабелю!

Интерфейс SCSI

Выпускается несколько комплектов микросхем, реализующих интерфейс SCSI, поэтому продавцы иногда ставят микросхемы поддержки SCSI прямо на материнскую плату. Стандарт SCSI определяет общий канал передачи данных, которым могут пользоваться периферийные устройства любых типов. В недавнем прошлом его использовали для подключения дисков, накопителей на магнитной ленте, сканеров и принтеров, но в настоящее время большинство изготовителей периферийных устройств отказались от интерфейса SCSI в пользу USB. Стандарт SCSI не налагает ограничений на конструкцию и формат жесткого диска, он лишь регламентирует взаимодействие диска с другими устройствами.

Стандарт SCSI прошел несколько стадий развития; текущая версия называется SCSI-3. Первоначальный вариант, SCSI-1, был разработан в 1986 г. на основе коммерческого интерфейса системной шины SASI (Shugart Associates System Interface) и принят в качестве стандарта ANSI. Вариант SCSI-2 появился в 1990 г. и был обратно совместим со спецификацией SCSI-1, но обладал дополнительными средствами повышения производительности. К таковым относились, в частности, очереди команд, позволявшие устройствам переупорядочивать запросы ввода-вывода для повышения производительности, и поддержка несмежного ввода-вывода, дававшая возможность осуществлять прямой доступ к несмежным областям памяти.

Существуют разновидности интерфейса SCSI-2, называемые "fast" (быстрая) и "wide" (расширенная), что означает, соответственно, удвоение скорости работы шины и увеличение числа битов, одновременно передаваемых по шине (обычно 16 или 32 вместо 8)[3]. Спецификация Wide SCSI допускает подключение до 16 устройств, тогда как в обычном варианте интерфейса — не более 8. Скорость и ширина канала — независимые характеристики, которые могут присутствовать одновременно.

---

[1] Действительно ли мы получим дисководы, которые смогут считывать и записывать данные со скоростью равной 600 Мбит/с? Нет, но даже сейчас шина больше не является ограничивающим фактором.

[2] Все верно: по ряду причин кабель питания по своей конструкции сложнее кабеля данных.

[3] 32-разрядные шины SCSI не очень распространены. В некоторых из них требуется несколько кабелей, обозначаемых как кабель А и кабель Б.

SCSI-3 в действительности представляет собой семейство стандартов. В него входят спецификации различных физических сред, включая традиционные параллельные шины и высокоскоростные последовательные интерфейсы, такие как Fibre Channel и IEEE 1394 (FireWire). В нем определены также наборы SCSI-команд и введены улучшения, связанные с автоматической конфигурацией устройств, поддержкой мультимедийных приложений и новых типов устройств.

Хотя работы над спецификацией SCSI-3 ведутся с 1993 г., они все еще продолжаются. Тем не менее, многие ее элементы уже нашли воплощение на рынке, часто под названием Ultra SCSI. Спецификация SCSI-3 охватывает стандарт SCSI-2, поэтому их можно считать совместимыми. Но помните: подключение старого устройства к новой шине может привести к замедлению работы всей системы. Кроме того, снижается максимальная длина кабельной системы.

В табл. 7.1 перечислены различные версии SCSI с указанием характеристик шины и длины кабеля.

Максимальная длина кабеля в несимметричных системах Ultra SCSI и Wide Ultra SCSI зависит от числа используемых устройств. В случае восьми устройств предельная длина составляет 1,5 м; если используются только четыре устройства, шину можно увеличить до трех метров. В стандарте Wide Ultra SCSI поддержка шестнадцати устройств обеспечивается только в режиме дифференцирования.

В SCSI-устройствах встречаются разъемные соединения различных типов. Они бывают как внешними, так и внутренними. В обычных SCSI-устройствах установлены 50-контактные разъемы, а в устройствах с расширенной разрядностью — 68-контактные. Внутренние устройства, как правило, подключаются с помощью 50-контактной "широкой" вилки или 68-контактной вилки "мини-микро", установленной на плоском кабеле. Внешние устройства подключаются посредством 50- или 68-контактных разъемов "мини-микро" высокой плотности.

Таблица 7.1. Эволюция стандарта SCSI

Версия	Частота (МГц)	Разрядность (биты)	Скорость (Мбайт/с)	Длина (м)	Длина в режиме дифференцирования (м)
SCSI-1	5	8	5	6	25
SCSI-2	5	8	5	6	25
Fast SCSI-2	10	8	10	3	25
Fast/wide SCSI-2	10	16	20	3	25
Ultra SCSI	20	8	20	1.5[а]	25
Wide Ultra SCSI[б]	20	16	40	1.5[а]	25
Wide Ultra2 SCSI[б]	40	16	80	–[в]	25 (ВВД)[г] 12 (НВД)
Wide Ultra3 SCSI[д]	80	16	160	–[в]	12 (НВД)

[а] Переменная; см. комментарии в тексте.

[б] Стандарты Wide Ultra SCSI и Wide Ultra2 SCSI иногда называют Fast-20 wide SCSI и Fast-40 wide SCSI соответственно.

[в] В этих версиях используется только дифференцирование.

[г] ВВД — это высоковольтное дифференцирование, а НВД — низковольтное. Первое использовалось в ранних версиях SCSI и не поддерживается в спецификациях, появившихся после Wide Ultra2 SCSI.

[д] Иногда стандарт Wide Ultra3 SCSI называют стандартом Ultra-160. Аналогично, стандарты, в которых использованы удвоенная и утроенная скорости передачи данных, соответственно называют Ultra-320 и Ultra-640. Большинство новых дисководов соответствует стандарту Ultra-320.

Существует также разъем SCA (Single Connector Attachment — единое разъемное соединение), применяемый в дисковых массивах "горячей" замены. Это 80-контактное универсальное разъемное соединение, через которое могут подаваться все необходимые устройству сигналы, в том числе сигналы подключения к шине, питания и конфигурирования.

На рис. 7.1 изображены наиболее распространенные разъемные соединения. Все соединители изображены со стороны разъема.

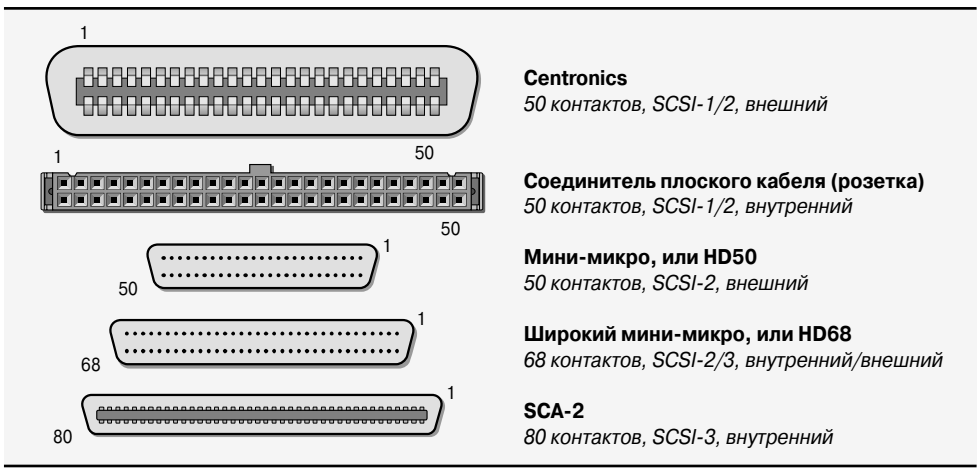


Рис. 7.1. Распространенные SCSI-разъемы (вид спереди; вилка, если не указано иное)

Шины SCSI организованы по принципу шлейфовой цепочки, поэтому у большинства внешних устройств есть два SCSI-порта. Оба порта абсолютно одинаковы и взаимозаменяемы, т.е. любой из них можно использовать как входной. По какой-то причине поставщики сканеров иногда считают, что эти правила на них не распространяются, и делают в устройстве всего один SCSI-порт. Соответственно, если внутренней оконечной нагрузки нет, то для таких устройств необходим особый тип терминатора.

Внутренние SCSI-устройства чаще всего подключаются к плоскому кабелю. Им нужен только один порт, поскольку разъемные соединения можно устанавливать прямо на плоский кабель. Необходимо убедиться в том, что контакт 1 шины SCSI подключен к контакту 1 жесткого диска (на плоском кабеле первый контакт обычно помечается красной полоской).

Оконечная нагрузка (терминатор) должна стоять на обоих концах шины SCSI. Терминаторы поглощают сигналы, достигающие конца шины, и предотвращают их отражение обратно в шину. Оконечные устройства различны по конструкции: от маленьких наружных вилок, вставляемых в обычный порт, до крошечных резисторных блоков, которые монтируются на печатных платах. Большинство современных устройств снабжено встроенными оконечными нагрузками.

Один конец шины обычно располагается внутри компьютера (на SCSI-контроллере либо на внутреннем SCSI-дисководе), а другой конец завершается на внешнем устройстве или SCSI-контроллере, если внешних устройств нет. Если на шине SCSI периодически возникают случайные ошибки, в первую очередь проверьте правильность оконечной нагрузки на обоих концах. Неправильное использование терминаторов — одна из

наиболее частых ошибок в SCSI-конфигурациях, причем ошибка не всегда очевидная и трудно обнаруживаемая.

Каждое SCSI-устройство имеет адрес, или "целевой номер", позволяющий отличать его от других устройств, подключенных к шине. Целевые номера располагаются в диапазоне от 0 до 7 или 15, в зависимости от типа шины: с обычной или расширенной разрядностью. Сам SCSI-контроллер тоже считается устройством и чаще всего имеет номер 7 (даже в шинах с расширенной разрядностью для обеспечения обратной совместимости). Целевые номера всех остальных устройств должны быть уникальными. Распространенная ошибка — забыть, что у SCSI-контроллера есть номер, и назначить этот номер другому устройству.

SCSI-адрес выбирается произвольно. С технической точки зрения он определяет приоритет устройства на шине, но на практике это не имеет особого значения. В одних системах диск с наименьшим номером выбирается в качестве загрузочного, в других — требуется, чтобы загрузочный диск имел целевой номер 0.

Можно считать большой удачей, если на добавляемом устройстве есть внешний переключатель для установки целевых номеров. Существуют и другие способы установки целевых номеров, в частности с помощью DIP-переключателей и перемычек. Если методика задания номера устройства не ясна, обратитесь к документации, прилагаемой к аппаратным средствам. Сегодня большинство аппаратных спецификаций можно найти на Web-сайтах производителей соответствующих устройств.

Стандарт SCSI поддерживает дополнительную форму адресации, реализуемую в виде номеров логических модулей. Каждый целевой адрес может содержать несколько логических модулей. Наиболее наглядный пример — дисковый массив с одним SCSI-контроллером. Однако в реальных конфигурациях логические модули используются редко. Когда встречается термин "номер модуля SCSI", можно считать, что речь идет о целевом номере. Если у SCSI-устройства есть всего один логический модуль, то по умолчанию ему обычно присваивается номер 0.

Шины SCSI довольно просто конфигурируются, но нужно соблюдать определенные правила.

- Во многих рабочих станциях есть встроенные SCSI-устройства. При подключении нового устройства проверьте перечень текущих устройств, прежде чем выполнять перезагрузку.
- Подключив новое SCSI-устройство и перезагрузившись, проверьте, все ли устройства выявлены ядром. Большинство драйверов SCSI не обнаруживает устройств, имеющих одинаковый SCSI-адрес — подобная конфигурация недопустима. Конфликты адресации устройств SCSI могут приводить к очень странному поведению устройств.
- Некоторые блоки расширения (коробки с источником питания и одним или несколькими SCSI-устройствами) имеют внутреннюю оконечную нагрузку. Если после них подключены внешние устройства, могут возникнуть проблемы. Всегда проверяйте наличие двух терминаторов и следите, чтобы они были расположены на концах шины.
- Иногда дисковый переключатель, используемый для выбора SCSI-адресов, устанавливают задом наперед. В таких случаях он будет, конечно, изменять SCSI-адрес, но совсем не так, как указано на переключателе.
- Рассчитывая длину шины SCSI-2, не забудьте учесть кабели внутри устройств и блоков расширения (а они могут быть весьма длинными). Помните также, что

максимально допустимая длина кабеля уменьшается при подключении старых SCSI-устройств к шине нового типа.

- Никогда не забывайте о том, что SCSI-контроллер имеет собственный SCSI-адрес!

Что лучше: SCSI или IDE?

В предыдущих изданиях этой книги мы безоговорочно отдавали предпочтение SCSI, когда речь шла о серверах. Однако с появлением дисководов SATA ситуация изменилась коренным образом. Интерфейс SCSI просто не предоставляет тех преимуществ, что раньше. Диски SATA успешно конкурируют (а в ряде случаев и превосходят их) с эквивалентными дисками SCSI практически всех категорий. В то же время устройства SATA значительно дешевле и более доступны.

Одно из преимуществ SCSI-дисков — наличие встроенного процессора в контроллере, что снимает значительную часть нагрузки с центрального процессора системы. В загруженных системах это может давать существенный выигрыш в производительности. Естественно, вам самим решать, окупает ли этот выигрыш значительные дополнительные затраты.

Однако иногда устанавливать SCSI-устройство желательно или даже необходимо.

- Когда требуется обеспечить максимально возможную производительность, выбирайте SCSI. Изготовители жестких дисков проводят политику разграничения возможностей IDE- и SCSI-устройств, поскольку это выгодно с маркетинговой точки зрения, так как позволяет разнообразить линии выпускаемых продуктов. Некоторые IDE-диски показывают более высокую пиковую производительность, чем их SCSI-аналоги, но средняя рабочая производительность почти всегда выше у SCSI.
- Серверы и многопользовательские системы требуют использования SCSI-устройств. Интерфейс SCSI незаменим в плане возможности эффективно обрабатывать одновременные запросы. В интенсивно эксплуатируемой системе замена IDE-диска SCSI-аналогом приведет к ощутимому росту производительности.
- Когда требуется подключать много устройств, выход опять один: SCSI. SCSI-устройства хорошо взаимодействуют с устройствами других типов, тогда как IDE-диски стараются захватить контроль над шиной.

7.2. Конструкция современного жесткого диска

Конструкция современного жесткого диска и терминология, используемая для обозначения его составных частей, приведены на рис. 7.2. Эта информация дана главным образом для того, чтобы читатели могли наглядно представить, как работает жесткий диск. Программам, обращающимся к диску, не нужно знать его физическое устройство.

Накопитель на жестких дисках состоит из вращающихся круглых пластин, покрытых магнитным слоем. Чтение и запись данных осуществляет маленькая головка, которая меняет ориентацию магнитных частиц на поверхности пластин. Корпус накопителя герметичен, чтобы внутрь не могли попасть ни пыль, ни грязь. Это делает жесткие диски гораздо более надежными, чем внешние носители.

На заре развития вычислительной техники в дисковых накопителях устанавливался, как правило, всего один диск. Рост емкости достигался за счет увеличения диаметра

пластины. На стене одной из наших лабораторий висит древний диск диаметром более четырех футов (120 см), который вмещал около 280 Кбайт данных.

В современных накопителях стоит не один большой диск, а несколько маленьких, размещенных этажеркой. Для хранения данных используются обе стороны пластин, хотя одна из них обычно содержит информацию для позиционирования, а не собственно данные. Одна пластина современного диска вмещает около 130 Гбайт данных.

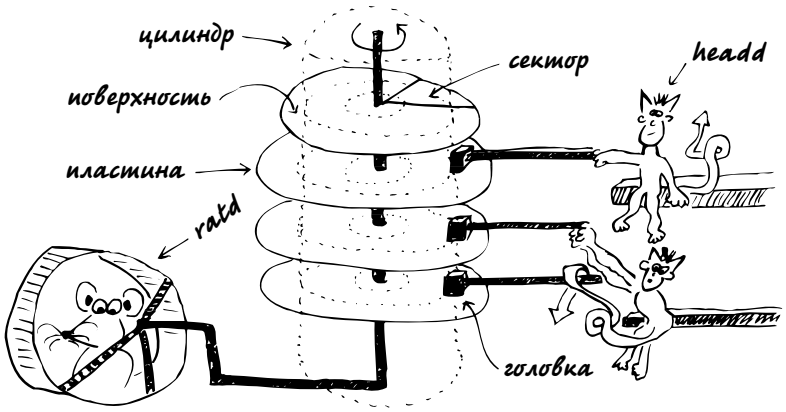


Рис. 7.2. Конструкция жесткого диска

Диски вращаются с постоянной скоростью. Данные читаются и записываются с помощью маленьких головок, которые скользят над поверхностями дисков вперед-назад, подобно игле звукоснимателя в проигрывателе грампластинок. Головка перемещается очень близко к поверхности диска, но не касается ее. Если произойдет касание, случится то, что называют аварией головки. Последствия такой аварии могут быть весьма разрушительными для данных.

Скорости вращения дисков за последние годы существенно возросли. Старые диски делали 3600 или 5400 об/мин. Сегодня стандартный показатель — 7200 об/мин, а диски со скоростями вращения 10000 и 15000 об/мин завоевывают все большую популярность в секторе устройств высокого класса. Чем выше скорость вращения, тем меньше время доступа головки к нужному участку диска и выше скорость передачи данных, но одновременно увеличивается нагрев диска в связи с повышенным выделением тепла. Современные высокоскоростные диски требуют хорошей вентиляции.

Для каждой поверхности нужна как минимум одна головка. В старых накопителях головкам приходилось "пробегать" огромные расстояния. Конструкция современных дисков более рациональна, а диаметр пластины продолжает сокращаться. Если двадцать лет назад стандартным считался диаметр 14 дюймов, десять лет назад — 5,25 дюйма, то сегодня он составляет 3,5 дюйма и меньше.

Перемещение головки в нужную позицию для чтения конкретных данных называется поиском. Каждому дискретному положению, которое может занимать головка, соответствует дорожка. Дорожки делятся на секторы, размер которых обычно составляет 512 байт.

Совокупность дорожек разных пластин, находящихся на одном расстоянии от оси шпинделя, называется цилиндром. Если все головки будут двигаться вместе, как это

обычно происходит, то данные, записанные в одном цилиндре, можно будет читать без дополнительных перемещений. Головки перемещаются достаточно быстро, но диски все равно вращаются гораздо быстрее, поэтому обращение к диску без перемещения головки в новую позицию происходит с максимальной скоростью.

7.3. Файловые системы Linux

Средства поддержки файловых систем в Linux стремительно развивались в последние годы, по мере того как ОС Linux вбирала в себя свойства целого ряда других операционных систем. Виртуальная файловая система (Virtual File System — VFS) ядра особенно удобна на персональных компьютерах, так как позволяет монтировать платформо-зависимые файловые системы, в частности пресловутую FAT, применявшуюся в Windows.

Файловые системы ext2fs и ext3fs

Вторая расширенная файловая система, известная как ext2fs, в течение долгого времени использовалась в Linux наиболее широко. Код системы ext2fs писался под Linux, но это не помешало разработчикам, Реми Карду (Remy Card), Теодору Тсо (Theodore Ts'o) и Стивену Твиди (Stephen Tweedie), заимствовать многие концепции из файловой системы FFS (Fast File System), реализованной Керком Мак-Кузиком (Kirk McKusick) и единомышленниками в 1984 г. для BSD-систем.

Третья расширенная файловая система, или ext3fs, является существенным дополнением к ext2fs. В большинстве современных дистрибутивов она используется по умолчанию. По сути, к существующему коду ext3fs были добавлены лишь средства журнальной регистрации. Казалось бы, простая модификация, но она повлекла за собой необычайное повышение надежности, причем для ее реализации не пришлось менять внутреннюю структуру ext2fs. Можно даже монтировать файловую систему типа ext3fs в формате ext2fs — будет всего лишь недоступен механизм журнальной регистрации.

Существующую файловую систему типа ext2fs можно привести к типу ext3fs с помощью команды `tune2fs`. Например, если файловая система находится в разделе `/dev/hda4`, то для ее преобразования потребуется такая команда:

```
# tune2fs -j /dev/hda4
```

Затем нужно модифицировать соответствующую запись в файле `/etc/fstab`, поменяв тип ext2 на ext3 (об этом пойдет речь в следующем подразделе).

Файловая система типа ext3fs выделяет отдельный участок диска под журнал операций. Перед выполнением любой операции записи вносимое изменение сначала регистрируется в журнале, и лишь после этого происходит модификация файловой системы. В случае сбоя предыдущее состояние файловой системы восстанавливается на основании записей журнала.

Благодаря журнальной регистрации время, затрачиваемое на проверку каждой файловой системы, сокращается примерно до одной секунды (об этом рассказывается в разделе 7.6). Если не учитывать аппаратные сбои, то можно считать, что состояние файловой системы типа ext3fs проверяется и восстанавливается практически мгновенно.

ext3fs — единственная файловая система, которая подробно описана в этой главе.

Файловая система ReiserFS

Файловая система ReiserFS, созданная Гансом Рейзером (Hans Reiser) — еще одна интенсивно развивающаяся файловая система Linux, которая по умолчанию используется в дистрибутиве SUSE. Как и в ext3fs, в ReiserFS ведется журнал операций, благодаря чему файловая система остается согласованной, несмотря на различные инциденты, например сбои системы или внезапные перезагрузки (что не редкость в портативных и настольных ПК). 4-я версия файловой системы ReiserFS появилась в августе 2004 г. и она по умолчанию включена в пакет Ubuntu. В дистрибутивах Fedora и SUSE пока попрежнему используются предыдущая версия этой файловой системы.

Помимо журнального механизма в Reiser4 реализован модульный интерфейс, который позволяет разработчикам приложений и системным администраторам определять способ обработки и защиты файлов. Это дополнение способствует повышению степени защищенности файлов в специализированных средах. ReiserFS финансируется управлением DARPA[4] и является единственной файловой системой с открытым исходным кодом, которая, по заявлениям ее разработчиков, удовлетворяет требованиям безопасности, предъявляемым министерством обороны.

Новые алгоритмы, примененные в файловой системе Reiser4, обеспечивают более эффективное использование дискового пространства, чем их предшественники. В других файловых системах (в том числе в предшествующих версиях ReiserFS) для распределения блоков дискового пространства используются алгоритмы с применением сбалансированного дерева. Хотя этот испытанный подход достаточно эффективен, как правило, он предполагает компромисс между производительностью и эффективностью использования дискового пространства. Алгоритм "перемежающегося дерева", используемый в файловой системе Reiser4, не ставит администратора перед необходимостью выбора.

Файловые системы XFS и JFS

Следующие две файловые системы, претендующие на завоевание ведущего положения в современных системах Linux — XFS компании SGI и JFS корпорации IBM. Обе эти системы поддерживаются современными ядрами и дистрибутивами и являются высокопроизводительными файловыми системами, в которых ведутся журналы операций. У обоих этих файловых систем есть свои приверженцы, но пока ни одна из них не находит столь широкого применения, как ext3fs или ReiserFS.

Этому трудно найти рациональное объяснение, но, похоже, основная проблема состоит в том, что по основным характеристикам эти файловые системы достаточно мало отличаются от ext3fs и ReiserFS. Точнее говоря, нюансы реализации различны, а каждая из файловых систем несколько превосходит другие по тем или иным параметрам (например, XFS обладает отличными средствами управления фрагментацией и экстентами). Вместе с тем, производительность основных конкурентов остается примерно одинаковой, а круг приложений, в которых та или иная файловая система действительно может "блеснуть", достаточно ограничен.

Более подробную информацию об этих файловых системах можно найти соответственно по адресам `jfs.sourceforge.net` (JFS) и `oss.sgi.com/projects/xfs` (XFS).

[4] Defense Advanced Research Projects Agency (Управление перспективных исследований МО США) — та же организация, которая финансировала создание Internet.

7.4. Процедура добавления жесткого диска

Процедура добавления нового диска включает следующие этапы:

- подключение дискового накопителя к компьютеру;
- форматирование диска при необходимости (обычно это не требуется);
- разбивка жесткого диска на разделы и присвоение им меток;
- создание файловых систем в разделах диска;
- монтирование файловых систем;
- настройка автоматического монтирования файловой системы;
- создание разделов подкачки.

Ниже в общих чертах рассмотрен каждый из перечисленных этапов. В разделе 7.7 детально расписана последовательность всех операций.

Подключение диска

Способ подключения диска зависит от его интерфейса. Если это IDE-диск, постарайтесь сделать так, чтобы к каждой шине был подсоединен только один диск. Проверьте ориентацию кабелей и правильность установок "главный/подчиненный" на каждом диске. Если это SCSI-диск, убедитесь, что на обоих концах шины присутствует оконечная нагрузка, длина кабеля в пределах нормы и целевой номер нового устройства не конфликтует с номерами имеющихся устройств. Особенности подключения дисков в каждой из систем описаны в соответствующих разделах в конце этой главы.

Чтобы получить доступ к новому диску, нужно предварительно создать для него файлы устройств в каталоге `/dev`. В Linux автоматически создаются файлы для наиболее часто встречающихся дисковых устройств. Подробнее об этом рассказывается в разделе 5.4.

Одной неосторожной операцией записи на диск можно за считанные секунды полностью разрушить файловую систему, поэтому права доступа к файлам дисковых устройств следует устанавливать очень осторожно. Мы рекомендуем предоставлять членам группы *`operator`* или *`disk`* право на чтение. Это позволяет операторам резервного копирования выполнять команду `dump` без прав суперпользователя и запрещает рядовым пользователям осуществлять прямой доступ к диску.

Форматирование диска

Компании-изготовители часто указывают емкость дисков без учета форматирования. Как правило, около 10% емкости диска уходит на маркировку его поверхностей, которая необходима для того, чтобы аппаратные и программные средства могли находить записанные там данные. Покупая диск, всегда оценивайте его емкость в "форматированных" байтах и соответствующим образом сравнивайте цены.

Размеры дисков указывают в мегабайтах, каждый из которых представляет собой миллион байт, в отличие от памяти, один мегабайт объема которой равен 2^{20} (1 048 576) байт. Разность составляет около 5%[5]. Поэтому при оценке и сравнении емкостей дисков следует обращать внимание на используемые единицы измерения.

[5] Конечно, префикс "мега" действительно означает "миллион", поэтому принятую практику указания объема памяти нельзя назвать совершенно неоправданной.

В процессе форматирования на диск записываются адресные данные и метки синхронизации для разграничения секторов. Кроме того, выявляются "дефектные" блоки, т.е. некачественные участки носителей, которые невозможно использовать для чтения или записи. В современных накопителях контроль дефектных блоков осуществляет контроллер диска; об этом не нужно беспокоиться ни пользователю, ни драйверу[6].

Все жесткие диски предварительно форматируются изготовителем, и качество такого форматирования обычно выше, чем то, которого можно достичь в реальных условиях. Лучше избегать низкоуровневого форматирования, если оно не требуется. Когда начинают возникать ошибки чтения или записи, в первую очередь проверьте кабели, терминаторы и номера устройств (для SCSI-дисков), так как появление таких ошибок не всегда означает наличие дефектных блоков. Если после этого диск по-прежнему работает неустойчиво, лучше заменить его, а не тратить много часов на форматирование.

IDE-диски, как правило, не предназначены для самостоятельного форматирования. Тем не менее, изготовители могут предлагать специальные программы форматирования, обычно работающие в Windows. Убедитесь, что такая программа работает с диском, и тщательно следуйте инструкциям разработчиков программы.

SCSI-диски форматируются автоматически по команде, посылаемой с компьютера. Способ выдачи такой команды зависит от системы. На персональном компьютере команду часто можно послать из BIOS-программы контроллера SCSI.

Существуют различные утилиты, позволяющие проверять целостность диска, записывая на него произвольные комбинации данных, а затем считывая их. Тщательные проверки требуют длительного времени (до нескольких часов), а достоверность их результатов, как правило, оставляет желать лучшего. Если только вы не подозреваете, что диск поврежден и не имеете возможности просто его заменить (или же располагаете достаточным временем), лучше избегать выполнения этих проверок. При необходимости можете спокойно оставить компьютер, который выполняет тесты, включенным на всю ночь. Не беспокойтесь по поводу "износа" диска из-за слишком интенсивной эксплуатации или активного тестирования. Диски рассчитаны на непрерывную круглосуточную работу.

Разбивка жесткого диска на разделы и присвоение им меток

После форматирования диска и маркировки дефектных блоков его необходимо разбить на фрагменты, называемые *разделами*. Разбивка на разделы позволяет рассматривать диск как группу независимых областей данных, а не как одно большое пространство блоков. Кроме того, она дает возможность скрывать системные области диска, в частности блоки начальной загрузки и саму таблицу разделов, от высокоуровневого программного обеспечения (например, от файловой системы). Ядро упрощает представление компоновки диска, а остальные программы работают с абстракциями, каковыми являются разделы.

Наличие разделов облегчает резервное копирование, не позволяет пользователям посягать на чужие области диска, повышает производительность и ограничивает потенциальный ущерб, наносимый неуправляемыми программами. Таблица разделов хранится

[6] Любые дефектные блоки, которые появятся после форматирования диска, не будут обрабатываться автоматически. Их использование может привести к ошибкам чтения или записи и даже к потере данных.

на диске в виде записи, называемой меткой. Метка обычно занимает первые несколько блоков диска. Ее содержимое может быть разным, но в целом метка хранит достаточно информации для того, чтобы операционная система могла загрузиться.

Разделы имеют четкие границы и не пересекаются. Однако файл `/dev/sda`, определенный в качестве файла блочного устройства диска, по существу представляет собой образ всего диска в целом. В результате команды пользовательского уровня получают возможность обращаться к диску "напрямую", через этот файл устройства. Например, пользовательский процесс может записать метку диска или скопировать ее содержимое на резервный диск с помощью команды `dd`. Конечно, этот специальный файл следует использовать очень осторожно, так как можно мгновенно испортить все разделы на диске.

Разработчики некоторых систем идут еще дальше и позволяют определять несколько перекрывающихся наборов разделов. Например, разделы 1, 2 и 3 делят диск одним способом, а разделы 4 и 5 — другим. Предполагается, что пользователь работает с одним набором согласованных разделов, а остальные просто игнорирует. На практике такая конфигурация приводит к ошибкам и повреждению данных.

В современных системах по сравнению с их предшественницами количество разделов сокращено, но большинство систем все равно имеет минимум два раздела.

- **Корневой раздел**. В нем содержатся все файлы, необходимые для перевода системы в однопользовательский режим. Дубликат этого раздела, хранящийся на другом диске, может пригодиться для восстановления системы в случае аварии.
- **Раздел подкачки**. В области подкачки хранятся страницы виртуальной памяти, если они не помещаются в физической памяти. В каждой системе должен присутствовать хотя бы один раздел подкачки.

Существуют разные мнения о том, какой способ разбивки дисков на разделы лучше. Приведем некоторые советы.

- Если есть несколько дисков, создайте на одном из них копию корневой файловой системы и проверьте, может ли система загружаться с этого диска.
- Наращивая память на компьютере, увеличивайте и размер раздела подкачки. Он должен быть достаточным для обеспечения нормальной работы системы. Подробнее о виртуальной памяти рассказывается в разделе 25.3.
- Разбивка области подкачки на несколько дисков способствует повышению производительности. Это касается и файловых систем: помещайте интенсивно используемые файловые системы на отдельные диски. Замечания по этому поводу приведены в разделе 25.3.
- Если планируется создавать резервные копии раздела, не делайте его размер больше, чем емкость резервного устройства. Об этом пойдет речь в разделе 9.1.
- Данные, которые часто подвергаются изменениям, целесообразно размещать в таких разделах, резервные копии которых создаются чаще.
- Помещение каталога временных файлов `/tmp` в отдельной файловой системе позволит отвести для них ограниченную область дискового пространства и не включать временные файлы в резервные копии.
- Поскольку журнальные файлы хранятся в каталоге `/var`, лучше сделать его отдельным разделом диска. Если же он будет частью очень маленького корневого раздела, то это может привести к быстрому заполнению последнего и "зависанию" системы.

- Каталог `/home` также целесообразно определять в качестве отдельного раздела. В этом случае даже при повреждении или уничтожении корневого раздела данные пользователя скорее всего останутся невредимыми. И наоборот, система сможет продолжить работу даже после того, как ошибочный пользовательский сценарий интерпретатора вызовет переполнение каталога `/home`.

Создание файловых систем в разделах диска

Даже после того как жесткий диск разбит на разделы, он еще не готов к хранению файлов. Свою лепту в подготовку диска к работе должна внести и файловая система.

Для инсталляции файловой системы в один из разделов диска предназначена команда `mke2fs -j`. (В Linux отсутствует команда `mke3fs`; опция `-j` указывает команде `mke2fs` о необходимости создания журнала файловой системы ext3fs.) В большинстве случаев достаточно лишь ввести команду `mke2fs -j` и указать имя раздела. Однако будьте внимательны при использовании этой команды; ошибка может привести к разрушению всей системы, а восстановление данных может оказаться крайне трудной, а то и невозможной задачей.

Файловая система типа ext3fs состоит из пяти структурных компонентов:

- набора блоков для индексных дескрипторов;
- набора распределенных “суперблоков”;
- таблицы распределения дисковых блоков;
- сводки использования блоков;
- совокупности блоков данных.

Дисковое пространство выделяется для журнала так, как если бы он был обычным файлом в корневом каталоге новой файловой системы, поэтому в действительности он не является отдельным структурным компонентом.

Каждый раздел файловой системы разбивается на группы блоков. Дисковое пространство для групп блоков таких структур, как таблицы индексных дескрипторов, выделяется таким образом, чтобы те блоки, обращения к которым взаимосвязано, могли храниться на диске один вблизи другого. Это группирование уменьшает потребность поиска по всему диску при обращении к блокам одного файла.

Индексные дескрипторы — это табличные записи фиксированной длины, которые содержат информацию об одном файле. Поскольку пространство для индексных дескрипторов выделяется при начальном структурировании файловой системы, необходимо заранее решить, сколько индексных дескрипторов следует создавать. Невозможно предсказать точно, сколько файлов (индексных дескрипторов) понадобится в будущем. Команды построения файловой системы используют эмпирическую формулу для оценки этого числа на основании величины раздела и среднего размера файла.

При создании файловой системы можно увеличить или уменьшить количество индексных дескрипторов. В файловых системах с множеством мелких файлов (например, в разделах, где хранятся исходные коды программ) должно быть больше индексных дескрипторов, а в файловых системах с несколькими большими файлами (например, в разделах баз данных) — меньше.

Суперблок — это запись, содержащая характеристики файловой системы. В суперблоке хранятся: информация о длине дискового блока, размере и расположении таблиц индексных дескрипторов; таблица распределения дисковых блоков и информация об их использовании, а также данные о размерах групп блоков и другие важные параметры

файловой системы. Поскольку повреждение суперблока приводит к удалению исключительно важной информации, создается несколько копий суперблока, которые размещаются в разных местах (в начале каждой группы блоков).

Для каждой монтируемой файловой системы ядро хранит одну копию суперблока в памяти и несколько копий на диске. Системный вызов `sync` периодически записывает кэшированные суперблоки на диск, восстанавливая целостность файловой системы. Это позволяет свести к минимуму ущерб, который может быть нанесен аварийным остановом компьютера. Вызов `sync`, кроме того, записывает на диск модифицированные индексные дескрипторы и кэшированные блоки данных. Традиционно синхронизация выполнялась с интервалами равными 30 с, но в файловой системе ext3fs она выполняется каждые 5 с.

Таблица распределения дисковых блоков содержит сведения о местонахождении свободных блоков в файловой системе. При создании новых файлов эта таблица анализируется на предмет выбора наиболее эффективной схемы размещения данных. Сводка использования блоков отражает основную информацию о блоках, которые уже используются.

Монтирование файловых систем

Чтобы файловая система стала доступной для процессов, ее нужно смонтировать. Точкой монтирования может стать любой каталог. Содержащиеся в нем файлы и подкаталоги будут недоступны, пока файловая система смонтирована “поверх них”. Подробнее о монтировании файловых систем рассказывалось в разделе 5.2.

После подключения нового диска необходимо смонтировать новые файловые системы вручную, чтобы убедиться в правильности их работы. Например, команда

```
# mount /dev/sda1 /mnt
```

монтирует файловую систему, находящуюся в разделе, который представлен файлом устройства `/dev/sda1`, в каталоге `/mnt`, традиционно используемом для временно монтируемых файловых систем. Если это новая файловая система, то ее содержимое будет выглядеть приблизительно так:

```
# ls /mnt
lost+found
```

Каталог `lost+found` автоматически создается при формировании файловой системы. Он используется командой `fsck` в экстренных случаях, поэтому не удаляйте его. В этом каталоге зарезервировано дополнительное место, чтобы команда `fsck` могла хранить там “несвязанные” файлы, не заботясь о выделении дисковой памяти в нестабильно работающей файловой системе.

Определить размер файловой системы позволяет команда `df`. Например:

```
$ df /home
Filesystem    1K-blocks      Used    Available      Use%      Mounted on
/dev/hda5       4128448    697968      3220768       18%      /home
```

Единицы измерения размеров блоков, которыми команда `df` оперирует по умолчанию — блоки по 1 Кбайт, но эту установку можно изменить с помощью опции `-h`, чтобы запросить вывод значений в более удобном для прочтения виде (например, 2,4 Гбайт).

Настройка автоматического монтирования файловой системы

Как правило, систему конфигурируют для монтирования локальных файловых систем автоматически на этапе начальной загрузки. Файл `/etc/fstab` содержит список устройств, которые соответствуют файловым системам.

Файл `/etc/fstab`, содержащий описание указанной выше файловой системы, может выглядеть так:

```
LABEL=/         /             ext3     defaults          1   1
none            /dev/pts      devpts   gid=5,mode=620    0   0
/dev/hda5       /home         ext3     defaults          1   2
none            /proc         proc     defaults          0   0
none            /dev/shm      tmpfs    defaults          0   0
/dev/cdrom      /media/cdrom  iso9660  ro,noauto,owner   0   0
/dev/hda3       /usr          ext3     defaults          1   2
/dev/hda6       /var          ext3     defaults          1   2
/dev/hda2       swap          swap     defaults          0   0
```

Каждая строка файла состоит из шести полей, разделенных пробелами, и содержит описание одной файловой системы. Для наглядности поля обычно выравнивают, но делать это необязательно.

О файловой системе NFS рассказывается в главе 16.

В первом поле указывается имя устройства или метка, закрепленная за ним с помощью команды `e2label` (поля вида `LABEL=`*`метка`*). Файл `/etc/fstab` может включать имена точек монтирования удаленных систем. В этом случае первое поле будет содержать путь к NFS-каталогу. Запись *`сервер:/экспорт`* обозначает каталог */`экспорт`* на компьютере *`сервер`*. Для файловых систем, которые не имеют реального постоянного места хранения (такие как `/proc` и `/dev/shm` в приведенном примере), поле устройства содержит метку-заполнитель `none`.

Во втором поле указывается точка монтирования, а в третьем — тип файловой системы. Тип локальных файловых систем обозначается по-разному, в зависимости от конфигурации системы. Для файловых систем ext*fs используются имена `ext2` and `ext3`, а для ReiserFS — `reiserfs`.

В четвертом поле перечислены опции монтирования. Ключевое слово `defaults` указывает комбинацию опций `rw`, `suid`, `dev`, `exec`, `auto`, `nouser` и `asyn` (эти опции подробно описаны на man-странице команды `mount`). Можно также добавить опцию `acl`, чтобы активизировать списки контроля доступа (см. раздел 5.6).

Режим файловой системы ext3fs задается с помощью записи `data=`*`режим`* файла `/etc/fstab`, где значение аргумента *`режим`* может быть таким: `ordered`, `writeback` или `journal`. *`Режим`* представляет режим работы, а не характеристику самой файловой системы. Поэтому он указывается в файле `fstab`, а не в качестве аргумента команды `mke2fs`.

Эти опции определяют три различных варианта компромисса между производительностью и надежностью:

- В стандартном, “упорядоченном” (`ordered`) режиме гарантируется, что файловая система всегда будет согласованной, и файлы не будут повреждаться в случае сбоев. В большинстве случаев это лучший выбор.

- В режиме "отложенной записи" (`writeback`) возможно повреждение файлов, но операции записи в файл в ряде случаев выполняются быстрее. Именно так работает большинство журнальных файловых систем, и это не намного рискованнее, чем стандартный режим ext2fsfs.
- В "журнальном" (`journal`) режиме используется журнал большего размера, что замедляет восстановление файловых систем при перезагрузке, но ускоряет работу с базами данных.

В пятом поле файла `fstab` приведена "частота" создания архивов. Теоретически это поле может использоваться программами резервного копирования, но обычно оно игнорируется.

Команда `fsck` описана в разделе 7.6.

В шестом поле указывается порядок проверки файловых систем командой `fsck`. Файловые системы с одинаковым значением этого поля проверяются параллельно, если это возможно. До появления журнальных файловых систем, когда выполнение команды `fsck` занимало значительное время, это поле имело большое значение, но теперь оно не столь важно. Нежелательно, чтобы одновременно проверялись не журнальные файловые системы, находящиеся на одном диске: головка диска будет так часто перемещаться вперед-назад, что производительность существенно снизится.

Файл `/etc/fstab` читают команды `mount`, `umount`, `swapon` и `fsck`, поэтому важно, чтобы представленные в нем данные были полными и достоверными. Команды `mount` и `umount` берут из этого файла недостающую информацию, если в командной строке указано только имя раздела или точка монтирования. Например, при наличии приведенного выше файла `fstab` команда

```
# mount /media/cdrom
```

будет интерпретирована так:

```
# mount -t iso9660 -o ro,noauto,owner /dev/cdrom /media/cdrom
```

Команда `mount -a` монтирует все "обычные" файловые системы, перечисленные в файле `fstab`. Она вызывается из сценариев запуска системы. Флаг `-t` ограничивает ее действие файловыми системами определенного типа. Например, команда

```
# mount -at ext3
```

монтирует все локальные файловые системы типа ext3fs. Команда `mount` читает файл `fstab` последовательно, поэтому записи файловых систем, монтируемых к другим файловым системам, должны следовать в файле за описаниями родительских разделов. Например, строка для точки монтирования `/var/log` должна находиться после строки для точки `/var`, если `/var` — отдельная файловая система.

Команда `umount`, предназначенная для размонтирования файловых систем, имеет похожий синтаксис. Размонтирование файловой системы, в которой какой-либо процесс хранит свой текущий каталог или в которой есть открытые файлы, запрещается. О том, как выйти из подобного положения, рассказывалось в разделе 5.2.

Создание раздела подкачки

Одно из преимуществ UNIX — использование концепции виртуальной памяти. Эта особенность позволяет операционной системе "делать вид", что у компьютера больше памяти, чем есть на самом деле. Когда процессы пытаются использовать "дополнитель-

ную" память, запросы переадресуются к жестким дискам, которые выступают в качестве очень медленного ОЗУ. Переброска содержимого памяти на диск и обратно называется страничным обменом. Было бы прекрасно, если бы система имела достаточный объем физической памяти, чтобы потребность в этом крайне медленном процессе никогда не возникала.

Как правило, в качестве области подкачки используются неупорядоченные разделы, а не структурированные файловые системы, что способствует повышению эффективности подкачки. В целях управления содержимым области подкачки ядро поддерживает собственную, упрощенную схему соответствия блоков памяти дисковым блокам. В качестве области подкачки можно использовать и обычный дисковый файл, а не специализированный раздел диска, но в этом случае страничный обмен будет осуществляться медленнее.

📖 Более подробно разбивка области подкачки рассматривается в разделе 25.3.

Чем больше раздел подкачки, тем больше виртуальной памяти доступно процессам. Максимальная эффективность виртуальной памяти достигается в случае распределения области подкачки между несколькими дисками (или, еще лучше, между несколькими шинами SCSI).

Разрешить страничный обмен с конкретным устройством можно вручную, но, как правило, лучше, чтобы эта функция выполнялась автоматически на этапе начальной загрузки. Области подкачки можно указывать в файле `/etc/fstab` — том самом, который содержит перечень монтируемых файловых систем. Строка с информацией об области подкачки выглядит так:

```
/dev/hda2      swap              swap     defaults            0   0
```

В процессе начальной загрузки выполняется команда `swapon`, которая разрешает страничный обмен для всех разделов подкачки, перечисленных в файле `fstab`.

7.5. HDPARM: УСТАНОВКА ПАРАМЕТРОВ ИНТЕРФЕЙСА IDE

Программа `hdparm` взаимодействует с IDE-драйвером Linux, извлекая и изменяя параметры диска. В числе выполняемых ею функций она может устанавливать параметры электропитания дисковода, включать и отключать прямой доступ к памяти, устанавливать флаг "только чтение" и выводить подробную информацию о дисководе. Настроив всего несколько из этих параметров диска, можно значительно повысить производительность всей системы. Эта утилита предназначена для работы исключительно с IDE-устройствами и не работает с дисководами SCSI или USB.

Она имеет следующий синтаксис:

```
hdparm [параметры] [устройство]
```

Запуск команды `hdparm` без параметров ведет к выводу информации о текущем состоянии некоторых настроек, представляющих наибольший интерес.

```
$ sudo /sbin/hdparm -d /dev/hdb

/dev/hdb:
  multcount    =  0 (off)
  IO_support   =  0 (default 16-bit)
  unmaskirq    =  0 (off)
  using_dma    =  0 (off)
```

```
keepsettings =  0 (off)
readahead    = 256 (on)
geometry     = 36481/255/63, sectors = 300069052416, start = 0
```

С точки зрения производительности диска эти настройки оставляют желать лучшего.

- Прямой доступ к памяти (DMA) отключен. Все современные диски могут пересылать данные непосредственно в память, минуя центральный процессор. Поэтому практически всегда целесообразно включать функцию DMA. Но иногда, при использовании неудачно спроектированного оборудования, это может приводить к аппаратным сбоям. Поэтому всегда вначале необходимо выполнить проверку в тестовой системе.
- Поддержка 32-разрядного ввода-вывода отключена. В этом случае данные передаются по шине PCI блоками по 16-разрядов, фактически снижая производительность вдвое.
- Отмена маскирования прерываний отключена. Включение этой функции (работа которой не совсем понятна) позволяет дисководу демаскировать ожидающие прерывания при обслуживании текущего прерывания диска.

Программа `hdparm` снабжена встроенными средствами тестирования производительности дисковода, которые помогают оценить влияние указанных параметров. В частности, при использовании этих заданных по умолчанию значений применительно к достаточно давно изготовленному дисководу IDE в нашей системе был получен следующий результат:

```
$ sudo /sbin/hdparm -Tt /dev/hdb

/dev/hdb:
 Timing cached reads:       228 MB in 2.41 seconds = 94.70 MB/sec
 Timing buffered disk reads:  6 MB in 4.62 seconds =  1.30 MB/sec
```

Запись `cached reads` (кэшированные чтения) указывает скорость передачи данных по шине IDE (не зависящую от пропускной способности физического дискового носителя), а запись `buffered reads` (буферизованные чтения) учитывает дополнительную нагрузку, связанную с чтением с физических пластин. Как и следовало ожидать, физический диск работает значительно медленнее шины. Но даже при этом скорость равная 1,3 Мбайт/с — весьма низка.

Поскольку функция прямого доступа к памяти — весьма важная характеристика современного оборудования, она заслуживает незамедлительной настройки. Наиболее современные дисководы поддерживают несколько режимов DMA. Скорости передачи данных при использовании различных режимов DMA для каждой из современных DMA-технологий приведены в табл. 7.2.

Таблица 7.2. Максимальные скорости передачи данных (в Мбайт/с) в различных режимах DMA

Режим	0	1	2	3	4	5	6
PIO	3,3	5,2	8,3	11,1	16,7		
SDMA	2,4	4,2	8,3				
MDMA	4,2	13,3	16,7				
UDMA	16,7	25,0	33,3	44,4	66,7	100,0	133,0

Эта таблица позволяет без труда определить соответствующий аргумент, который должен быть передан команде hdparm. DMA для UDMA2 можно включить с помощью следующей команды hdparm:

```
$ sudo /sbin/hdparm -d1 -Xudma2 /dev/hdb

/dev/hdb:
  setting using_dma to 1 (on)
  setting xfermode to 66 (UltraDMA mode2)
  using_dma = 1 (on)
```

Необходимо включить поддержку 32-разрядного ввода-вывода:

```
$ sudo /sbin/hdparm -c1 /dev/hdb

/dev/hdb:
  setting 32-bit IO_support flag to 1
  IO_support = 1 (32-bit)
```

И, наконец, значение демаскирования прерываний является двоичным. Мы включим его с помощью команды

```
$ sudo /sbin/hdparm -u1 /dev/hdb

/dev/hdb:
  setting unmaskirq to 1 (on)
  unmaskirq = 1 (on)
```

Влияние этих изменений можно оценить, снова используя средства сравнения производительности программы hdparm:

```
$ sudo /sbin/hdparm -Tt /dev/hdb

/dev/hdb:
  Timing cached reads:       256 MB in   2.00 seconds = 127.83 MB/sec
  Timing buffered disk reads: 40 MB in   3.01 seconds =  13.30 MB/sec
```

Как видите, всего несколько изменений в конфигурации hdparm привели к значительному повышению обоих значений.

Поэкспериментируйте с различными значениями, чтобы найти оптимальные параметры настройки. Чтобы получить наиболее точные данные, имеет смысл несколько раз запустить тест производительности для каждого параметра.

7.6. Команда fsck: проверка и восстановление файловых систем

Современные файловые системы на удивление надежны, и даже реализации, которые не поддерживают ведение журнала операций, успешно справляются с неожиданными сбоями системы и отказами аппаратных средств. В то же время ряд факторов могут привести к повреждению или нарушению целостности файловых систем.

Всякий раз когда ядро выдает “паническое” сообщение или происходит сбой питания, в тех файловых системах, которые непосредственно перед сбоем находились в активном состоянии, происходят незначительные нарушения. Поскольку ядро хранит в памяти как блоки данных, так и сводку использования блоков, самый последний об-

раз файловой системы распределяется между диском и памятью. При сбое информация, хранящаяся в памяти, утрачивается. Хранящиеся в памяти блоки замещаются теми их версиями, которые были последними записаны на диск.

Существует несколько подходов к решению этой проблемы. Небольшие повреждения обычно поддаются устранению с помощью команды `fsck` (сокращение от "filesystem consistency check" — проверка целостности файловой системы). С концептуальной точки зрения это не очень элегантный подход, но он хорошо подходит для большинства распространенных ситуаций.

Файловые системы, в которых ведется журнал операций, например ReiserFS и ext3fs, записывают метаданные в последовательный журнальный файл, который сохраняется на диске перед завершением каждой операции записи. Впоследствии метаданные переносятся из журнального файла в соответствующее место файловой системы. Если происходит сбой системы, журнальный файл просматривается в обратном порядке и находится точка, когда система последний раз пребывала в согласованном состоянии. В результате не требуется полная проверка файловой системы. Команда `fsck` по-прежнему запускается на этапе начальной загрузки для проверки того, что файловая система находится в согласованном состоянии, но теперь эта команда выполняется гораздо быстрее, чем в случае файловой системы ext2fs. Эффект тем сильнее, чем больше размер файловой системы.

Если журнал операций не поддерживается, остается надеяться на команду `fsck`. Ниже перечислены пять наиболее часто встречающихся ошибок:

- наличие индексных дескрипторов, на которые нет ссылок;
- неоправданно большие значения счетчиков ссылок;
- неиспользуемые блоки данных, не отраженные в таблицах блоков;
- блоки данных, помеченные как свободные, но используемые в файлах;
- неверная статистическая информация в суперблоке.

Команда `fsck` устраняет все эти проблемы автоматически. Если в файловую систему вносятся изменения, команду нужно повторно выполнять до тех пор, пока файловая система не станет согласованной.

Диски обычно проверяются на этапе начальной загрузки с помощью команды `fsck -p`, которая исследует локальные файловые системы, перечисленные в файле `/etc/fstab`, и исправляет пять вышеуказанных ошибок. Linux запоминает, какие файловые системы не были размонтированы корректно, и проверяет только их. Если ведется журнал операций, команда `fsck` сообщает о том, что поддерживается журнальный режим и восстанавливает последнее согласованное состояние файловой системы на основании записей журнала.

Команду `fsck -p` можно выполнить и для отдельной файловой системы, например:

```
# fsck -p /dev/sda5
```

Когда команда `fsck` читает файл `fstab`, чтобы узнать, какие файловые системы необходимо проверить, она следует указаниям, оставленным для нее в последнем поле каждой записи. Файловые системы проверяются в порядке возрастания номеров, приведенных в последнем поле. Если две файловые системы находятся на разных дисках, им можно присвоить одинаковый порядковый номер. Команда `fsck` проверит их одновременно, т.е. сократится время ожидания дискового ввода-вывода. Первым всегда следует проверять корневой раздел.

📖 Для ознакомления с некоторыми (не столь полезными) опциями обратитесь к странице интерактивного руководства команды `tune2fs`.

Систему Linux можно сконфигурировать так, чтобы проверка файловой системы выполнялась после выполнения определенного числа операций монтирования, даже если все операции демонтажа были выполнены корректно. Эта предосторожность служит хорошим средством профилактики, и в большинстве случаев стандартное значение (обычно равное 20) вполне приемлемо. Однако в системах, в которых монтирование файловых систем выполняется часто (например, в настольных рабочих станциях), выполнение команды `fscks` после каждых 20 операций монтирования может оказаться обременительным. Чтобы увеличить интервал до 50 операций монтирования, необходимо использовать команду `tune2fs`:

```
$ sudo /sbin/tune2fs -c 50 /dev/sda3
tune2fs 1.35 (28-Feb-2004)
Setting maximal mount count to 50
```

Ошибки, не относящиеся ни к одной из вышеуказанных пяти категорий, потенциально опасны. Они заставляют команду `fsck -p` вывести сообщение о помощи и завершить свою работу. В этом случае выполните команду `fsck` без опции `-p`. При работе в ручном режиме команда запрашивает подтверждение на каждую восстановительную операцию, которую собирается выполнить. Вот некоторые ошибки, которые команда `fsck` считает опасными и требующими вмешательства человека:

- блоки, принадлежащие более чем одному файлу;
- блоки, находящиеся вне диапазона файловой системы;
- слишком маленькие счетчики ссылок;
- неучтенные блоки;
- каталоги, содержащие ссылки на невыделенные индексные дескрипторы;
- различные ошибки формата файлов.

К сожалению, не имея глубоких знаний о структуре файловой системы, восстановить диск вручную невозможно. Никогда не пытайтесь осуществлять прямую запись в файловую систему через файлы устройств.

На практике это означает, что выбор сводится к принятию исправлений, предлагаемых командой `fsck`. Проблемы можно минимизировать путем тщательной регистрации сообщений, выдаваемых командой, так как иногда в них содержится информация о сбойном файле (или файлах). Если команда запрашивает разрешение на удаление файла, попробуйте сначала скопировать его в другую файловую систему и только после этого разрешайте команде продолжить работу. Но помните: пытаясь обратиться к поврежденной файловой системе, вы рискуете вызвать фатальную ошибку ядра.

📖 О способах создания резервных копий рассказывается в главе 9.

Если поврежденная файловая система содержит ценные данные, а команда `fsck` не смогла автоматически ее восстановить, не экспериментируйте с ней, не создав предварительно резервную копию. Можно попробовать применить к диску команду `dump`, но она предполагает наличие неповрежденной файловой системы, поэтому полученный образ может оказаться неполным (или команда завершится выдачей сообщения об ошибке). Лучше всего перестраховаться и выполнить для всего диска команду `dd`, чтобы создать резервный диск.

Когда команда `fsck` обнаруживает файл, родительский каталог которого определить невозможно, она помещает такой файл в каталог `lost+found`, находящийся на верхнем уровне файловой системы. Поскольку имя, присвоенное файлу, регистрируется только в его родительском каталоге, то имена "потерянных" файлов определить невозможно, и файлам, помещаемым в каталог `lost+found`, присваиваются имена, совпадающие с номерами их индексных дескрипторов. Таблица индексных дескрипторов содержит идентификатор пользователя (UID) владельца файла, поэтому повторное присвоение файла его первоначальному владельцу не представляет особой сложности.

7.7. Добавление жесткого диска: пошаговое руководство

В этом разделе мы подробно рассмотрим процедуру конфигурирования нового дисковода. Диск будет содержать несколько разделов, в том числе один раздел подкачки. В остальных разделах мы создадим файловые системы типа ext3fs.

Подключив новый диск, убедитесь, что система распознает устройство, и только затем приступайте к загрузке ядра. Если это IDE-диск, проверьте, обнаруживается ли он программой конфигурации BIOS, которую можно вызывать нажатием специальной комбинации клавиш перед началом загрузки системы. Обратитесь к руководству, поставляемому с компьютером или материнской платой, чтобы получить информацию о конфигурировании BIOS для IDE-устройств. В большинстве случаев дополнительное конфигурирование не требуется.

Многие SCSI-платы тоже имеют программу настройки BIOS, которую можно вызвать перед началом загрузки системы. В этом случае необходимо опросить SCSI-шину, чтобы проверить, подключено ли к ней новое устройство. Если эта процедура приводит к зависанию или выдаче сообщения об ошибке, то, возможно, был выбран идентификатор устройства, который уже занят, или неправильно установлена оконечная нагрузка.

С помощью BIOS-программы SCSI-контроллера можно осуществить низкоуровневое форматирование диска. Для некоторых дисков эта операция занимает много времени, к тому же ее нельзя прервать, так что наберитесь терпения.

📖 Об инсталляции драйверов устройств рассказывается в разделе 28.6.

Если SCSI-плата не располагает пользовательскими средствами конфигурирования, всегда можно, загрузив систему, посмотреть сообщения, выдаваемые ядром. При отсутствии сообщений от SCSI-драйвера, скорее всего, потребуется инсталлировать драйвер, прежде чем устройство сможет быть распознано ядром.

В нашем случае SCSI-контроллер BusLogic выдал следующие сообщения:

```
scsi0 : BusLogic BT-948
scsi : 1 host.
  Vendor:  SEAGATE   Model: ST446452W      Rev: 0001
  Type:    Direct-Access                   ANSI SCSI revision: 02
Detected scsi disk sda at scsi0, channel 0, id 3, lun 0
scsi0: Target 3: Queue Depth 28, Asynchronous
SCSI device sda: hdwr sector=512 bytes. Sectors=91923356 [44884 MB] [44.9 GB]
sda: unknown partition table
```

Не обращайте внимания на сообщение о таблице разделов, так как это первое обращение к диску. Когда система загрузится, можно приступать к разбивке диска на разделы.

В первую очередь необходимо проверить, существуют ли файлы устройств для диска. В Linux имена файлов, соответствующих SCSI-дискам, имеют формат `/dev/sd`*`XN`*, где *`X`* — символ в нижнем регистре, идентифицирующий диск (`a` — SCSI-диск с наименьшим номером, `b` — следующий диск и т.д.[7]), а *`N`* — номер раздела, начинающийся с единицы. При ссылке на весь диск номер раздела просто опускается. В Linux нет байт-ориентированных (символьных) дисковых устройств.

В приведенном примере диск является первым в SCSI-цепочке. Следовательно, первый раздел будет называться `/dev/sda1`, а ссылка на весь диск будет иметь вид `/dev/sda`. Если файлов устройств по какой-то причине нет, их можно создать с помощью сценария `MAKEDEV` или команды `mknod`.

Теперь диск готов к разбивке на разделы. Как и в большинстве операционных систем на персональных компьютерах, в Linux соответствующая утилита называется `fdisk`. Хотя все версии этой утилиты работают примерно одинаково (они реализуют стандартную систему разбивки на разделы, предложенную компанией Microsoft), существует много отличий. Прочтите документацию и убедитесь, что имеющаяся версия утилиты `fdisk` работает так, как описано ниже.

```
# fdisk /dev/sda
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF
disklabel Building a new DOS disklabel. Changes will remain in memory only,
until you decide to write them. After that, of course, the previous content
won't be recoverable.

The number of cylinders for this disk is set to 5721.
There is nothing wrong with that, but this is larger than 1024,
and could in certain setups cause problems with:
1. software that runs at boot time (e.g., LILO)
2. booting and partitioning software from other OSs (e.g., DOS FDISK, OS/2 FDISK)
```

Поскольку предполагается работать с диском только в среде Linux, мы проигнорируем предупреждающее сообщение. В прошлом иногда важно было делать первый раздел маленьким, чтобы не возникало проблем со старыми BIOS-программами и другими операционными системами, установленными на том же диске.

Утилита `fdisk` является интерактивной. Введите `m`, чтобы отобразить список всех ее инструкций. Мы будем использовать следующие инструкции:

- `n` или `new` — создает новый раздел;
- `t` или `type` — изменяет тип раздела;
- `p` или `print` — отображает таблицу разделов;
- `w` или `write` — записывает таблицу разделов на диск.

Так как на нашем диске еще нет разделов, начнем с создания раздела. Если на диске имеются старые разделы, их нужно предварительно удалить с помощью команды `delete` утилиты `fdisk`. Утилита ничего не поменяет на диске, пока не будет дана команда записать таблицу разделов.

В таблице разделов есть место для четырех первичных разделов. Любой из них можно сделать расширенным, т.е. указывающим на другую таблицу разделов с еще четырьмя элементами — логическими разделами. Благодаря этому можно создавать более четырех разделов, но проще всего ограничиться только первичными разделами, что мы и сделаем:

[7] Обратите внимание на то, что эти буквы обозначают порядок целевых номеров SCSI-устройств, а не сами номера. Если добавить или удалить диск, поменяются все буквы!

```
Command (m for help): new Command action
   e      extended
   p      primary partition (1-4): p
Partition number (1-4): 1
First cylinder (1-5721, default 1): 1
Last cylinder or +size or +sizeM or +sizeK (1-5721, default 5721): +2G

Command (m for help): print
Disk /dev/sda: 255 heads, 63 sectors, 5721 cylinders
Units = cylinders of 16065 * 512 bytes
   Device    Boot    Start     End      Blocks    Id   System
/dev/sda1              1     255      2048256    83   Linux
```

Раздел подкачки создается аналогичным образом, только тип раздела будет `Swap`, а не `Linux`. Хотя ядру безразличен тип раздела, некоторые программы и сценарии пытаются на его основании понять назначение раздела. Мы указали размер этого раздела равным 2 Гбайт, что, возможно, чересчур много для большинства приложений, но раз уж у нас такой большой диск, можно позволить себе быть столь щедрыми. Если команда `mkswap` не сможет использовать все выделенное пространство, она выдаст предупреждение.

```
Command (m for help): new
   e    extended
   p    primary partition (1-4): p
Partition number (1-4): 2
First cylinder (256-5721, default 256): 256
Last cylinder or +size or +sizeM or +sizeK (256-1275, default 1275): 511
Command (m for help): type
Partition number (1-4): 2
Hex code (type L to list codes): 82
Changed system type of partition 2 to 82 (Linux swap)
```

Третий раздел создается аналогичным образом. Прежде чем записать таблицу разделов на диск, просмотрим ее:

```
Command (m for help): print
Disk /dev/sda: 255 heads, 63 sectors, 5721 cylinders
Units = cylinders of 16065 * 512 bytes
    Device  Boot   Start    End     Blocks   Id  System
/dev/sda1            1     255    2048256   83  Linux
/dev/sda2          256     511    2056320   82  Linux swap
/dev/sda3          512    5721   41849325   83  Linux
```

Рядом с количеством блоков появится звездочка, если раздел не заканчивается на границе цилиндра. В этом случае можно либо удалить раздел и создать его заново, указав нужное число цилиндров, либо смириться с тем, что небольшая часть диска будет недоступна. В данном случае с таблицей разделов никаких проблем нет, поэтому просто записываем ее на диск:

```
Command (m for help): write
The partition table has been altered!
Calling ioctl() to re-read partition table.
Syncing disks.
```

Теперь можно приступать к формированию файловых систем. Для создания файловой системы типа ext3fs нужно ввести команду `mke2fs -j`, указав имя устройства в качестве аргумента:

```
# mke2fs -j /dev/sda1
mke2fs 1.36 (05-Feb-2005)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
245280 inodes, 489974 blocks
...
Superblock backups stored on blocks:
   32768, 98304, 163840, 229376, 294912
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

This filesystem will be automatically checked every 34 mounts or 180 days,
whichever comes first. Use tune2fs -c or -i to override.
done
```

С помощью опции `-J` можно явно указать либо размер журнала, который будет расположен в новой файловой системе (`-J size=`*x*), либо внешнее устройство, содержащее журнал (`-J device=`*y*). В типичном случае журнал располагается внутри самой файловой системы и занимает от 1024 до 102 400 блоков.

Более крупная файловая система создается аналогично, но весь процесс длится значительно дольше. Если заранее известно, что не все индексные дескрипторы, создаваемые командой `mke2fs`, понадобятся, можно уменьшить их число, увеличив тем самым пространство для реальных данных. Точно так же имеет смысл увеличить количество индексных дескрипторов для файловой системы, в которой предполагается хранить множество очень маленьких файлов. Лучше иметь запас, поскольку при нехватке дескрипторов нельзя будет создать ни одного файла. Тем более что после создания файловой системы увеличить количество индексных дескрипторов нельзя. Если дескрипторов все же недостаточно, придется с помощью команды `dump` переписать образ файловой системы на ленту или на другой раздел, затем выполнить команду `mke2fs -i`, указав нужное число индексных дескрипторов, и, наконец, командой `restore` восстановить данные раздела. Малоприятное занятие, что и говорить!

После создания файловых систем нужно выполнить команду `fsck`, чтобы убедиться в их корректности. Флаг `-f` заставит команду проверять новые файловые системы, а не предполагать, что они пусты.

```
# fsck -f /dev/sda1 fsck 1.36 (05-Feb-2005) e2fsck 1.36 (05-Feb-2005)
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/sda1: 11/245280 files (9.1% non-contiguous), 16629/489974 blocks
```

Создав точки монтирования, можно приступать к монтированию файловых систем:

```
# mkdir /bkroot
# mount /dev/sda1 /bkroot
# df /bkroot
Filesystem      1k-blocks    Used   Available   Use%   Mounted on
/dev/sda1         1981000      13     1878575     0%   /bkroot
```

Чтобы обеспечить автоматическое монтирование новых файловых систем на этапе начальной загрузки, следует для каждой из них добавить запись в файл `/etc/fstab`. Каждая строка содержит информацию с именем устройства, точкой монтирования, типом файловой системы, опциями монтирования, частотой создания резервных копий и числом запусков команды `fsck`.

```
/dev/sda1       /bkroot           ext3     defaults          0   1
/dev/sda3       /new              ext3     defaults          0   2
```

Для того чтобы диск стал загрузочным, на него необходимо поместить программу начальной загрузки. Это делается с помощью команды `lilo` либо `grub`, в зависимости от системы. Подробнее об инсталляции и конфигурировании программ начальной загрузки рассказывалось в разделе 2.3.

Последний шаг заключается в создании области подкачки и добавлении ее к системе. Разделы подкачки инициализируются командой `mkswap`, которая в качестве аргумента принимает имя устройства. В последних версиях указание размера раздела подкачки не обязательно. Когда область подкачки создана, ее нужно активизировать с помощью команды `swapon`. Эта команда также проверяет правильность инициализации области.

```
# mkswap /dev/sda2
Setting up swapspace version 1, size = 2105667584 bytes
# swapon /dev/sda2
# swapon -s
Filename   Type          Size          Used     Priority
/dev/hda5  partition     133020        688      -1
/dev/sda2  partition     2056316       0        -2
```

Как всегда, следует добавить запись о новом разделе в файл `/etc/fstab`, чтобы система автоматически подключала его при перезагрузке. В нашем случае это будет такая запись:

```
/dev/sda2       swap              swap     defaults          0   0
```

И, наконец, необходимо, перезагрузившись, убедиться в том, что все файловые системы и раздел подкачки функционируют нормально.

7.8. Расширенное управление дисками: RAID и LVM

Рассмотренные нами процедуры добавления дисков, файловых систем и областей подкачки аналогичны используемым в большинстве систем UNIX. Однако Linux предоставляет ряд дополнительных возможностей, о которых пользователи многих других операционных систем все еще могут только мечтать. Две различные технологии — программная реализация RAID и управления логическими томами — придают управлению дисками в среде Linux дополнительный уровень гибкости и надежности.

В работе жестких часто возникают сбои, и даже при использовании последних резервных копий последствия сбоя диска для сервера могут оказаться катастрофическими. RAID (Redundant Array of Independent Disks — избыточный массив независимых дисков) — система, в которой данные распределяются или дублируются на нескольких жестких дисках. Применение RAID не только способствует предотвращению потери данных, но и минимизирует время простоя, связанное с аппаратными отказами (часто сводя его к нулю), а также теоретически способствует повышению производительности

системы. Системы RAID могут быть реализованы аппаратно, но в системе Linux все необходимые компоненты реализованы программно.

Второе, не менее полезное средство, получившее название LVM (logical volume management — управление логическими томами), помогает администраторам эффективно распределять дисковое пространство между разделами. Представьте себе ситуацию, когда точный размер создаваемого раздела не известен. Через полгода после создания раздела выясняется, что он слишком велик, а соседний раздел слишком мал... Знакомая ситуация, не правда ли? LVM позволяет динамически перераспределять дисковое пространство между разделами.

Хотя каждое из названных средств и само по себе предоставляет очень большие возможности, они особенно эффективны при совместном использовании. В последующих разделах описаны принципы построения обоих систем и приведены подробные примеры их конфигурирования.

Программная реализация RAID в среде Linux

Недавно мы столкнулись с отказом контроллера жесткого диска на важном рабочем сервере. Хотя данные были продублированы на нескольких физических дисках, отказ аппаратного контроллера RAID привел к уничтожению данных на всех дисках. Нам пришлось прибегнуть продолжительному и трудоемкому процессу восстановления данных с магнитной ленты, и для полного восстановления сервера потребовалось более двух месяцев. Теперь в нашем перестроенном сервере управление средой RAID осуществляется посредством программы ядра, что полностью исключает возможность еще одного отказа контроллера RAID.

RAID может выполнять две основные функции. Во-первых, он может способствовать повышению производительности, “разбивая” данные между несколькими дисками и тем самым позволяя нескольким дискам работать одновременно при чтении или записи одного потока данных. Во-вторых, он может дублировать или “отражать” данные на нескольких дисках, тем самым снижая риск, связанный с отказом одного единственного диска. RAID-системы Linux несколько отличается от традиционного, но и он логически делится на несколько уровней.

- Линейный режим не обеспечивает никакой избыточности данных или повышения производительности. Он просто объединяет блочные адреса нескольких дисков, создавая единый (и больший по размеру) виртуальный диск.
- RAID уровня 0 используется исключительно для повышения производительности. В этом массиве два или более дисков равного размера используются для уменьшения времени записи и доступа к диску.
- Массив RAID уровня 1 — первый уровень, обеспечивающий избыточность. Данные дублируются одновременно на двух или более дисках. В этом режиме обеспечивается зеркальное дублирование данных, но производительность снижается, поскольку информация должна записываться более одного раза.
- Массив RAID уровня 4 соперничает с RAID уровня 5 (уступая ему в плане поддержания целостности данных). В этом режиме выполняется разбиение данных, но один диск выделяется для записи информации четности, что влечет за собой ожидание при записи на этот диск. Поэтому, если только не существуют какие-либо веские причины для его использования, мы рекомендуем отказываться от RAID 4 в пользу RAID 5.

- RAID уровня 5 — наиболее эффективный тип RAID. Разбиение как данных, так и информации четности создает избыточную архитектуру с одновременным снижением времени чтения и записи. Для реализации RAID 5 требует использования не менее трех дисков.

Программная реализация RAID была встроена в ядро Linux еще с версии 2.0, но первые версии содержали ошибки и были неполными. Мы рекомендуем избегать использования реализаций, предшествующих встроенным в ядро версии 2.4.

Управление логическими томами

LVM (Logical Volume Manager — Диспетчер логических томов) — это дополнительная подсистема, которая реализует особый вид разбивки на разделы. Диспетчер позволяет объединять отдельные диски в "группы томов". Суммарное пространство, занимаемое группой и рассматриваемое как единое целое, делится на логические тома, доступ к которым осуществляется как к обычным блочным устройствам. Благодаря этому становятся возможными:

- более эффективное управление дисковым пространством;
- перемещение логических томов между различными физическими устройствами;
- динамическое изменение размеров логических томов;
- создание "мгновенных" копий целых файловых систем;
- оперативная замена дисков без останова системы.

Компоненты логического тома можно объединять различными способами. В режиме конкатенации физические блоки каждого диска остаются неразрывными, а диски как бы "перетекают" друг в друга. В режиме чередования соседние блоки виртуального диска могут на самом деле находиться на разных физических устройствах. Часто это позволяет повысить пропускную способность и уменьшить время, затрачиваемое на поиск данных.

Пример конфигурации, использующей LVM и RAID

В предыдущем размере мы рассмотрели конфигурацию основного жесткого диска. В этом разделе мы рассмотрим процедуру настройки системы, которая включает в себя как RAID, так и LVM. Подобная конфигурация может быть особенно полезна для производственных серверов.

Мы создадим массив RAID 5 из трех пустых дисков. В этом массиве RAID будут определены два раздела LVM — `web1` и `web2`. Описанная структура предоставляет несколько преимуществ по сравнению с традиционной системой.

- RAID 5 обеспечивает избыточность данных; отказ одного диска не влечет за собой потерю данных. В отличие от RAID 4, в этом массиве не имеет значения, какой именно из дисков отказывает!
- LVM позволяет динамически изменять размеры логических разделов. Когда энергичный Web-мастер заполнит раздел `web2`, можно без труда заимствовать часть дополнительного дискового пространства в разделе `web1`.
- Со временем дополнительное дисковое пространство может потребоваться обоим логическим разделам. Структура массива RAID 5 допускает добавление в него дополнительных дисков. После этого существующие группы LVM могут быть расширены на дополнительное дисковое пространство без повторного создания каких-либо разделов.

Сперва мы рассмотрим начальную конфигурацию, а затем опишем, как следует поступать с отказавшим диском, и приведем методику изменения размера раздела LVM.

Описываемая система состоит из четырех дисков SCSI равных размеров:

```
# fdisk -l

Disk /dev/sda: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device    Boot   Start         End    Blocks       Id        System
/dev/sda1       *       1          13    104391       83        Linux
/dev/sda2      14     144     1052257+        82       Linux     swap
/dev/sda3     145    2213    16619242+        8e       Linux     LVM1G

Disk /dev/sdb: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device    Boot    Start         End    Blocks       Id        System

Disk /dev/sdc: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device    Boot    Start         End    Blocks       Id        System

Disk /dev/sdd: 18.2 GB, 18210036736 bytes
255 heads, 63 sectors/track, 2213 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device    Boot    Start         End    Blocks       Id        System
```

Первый диск SCSI, `/dev/sda`, содержит разделы системы. Остальные три диска (`sdb`, `sdc` и `sdd`) не содержат таблиц логических разделов.

Прежде всего, создадим логические разделы на каждом из дисков SCSI. Поскольку диски одинаковы, для каждого из них можно выполнить один и тот же набор команд.

```
# fdisk /dev/sdb
...
Command (m for help): new
Command action
   e   extended
   p   primary partition (1-4): p
Partition number (1-4): 1
First cylinder (1-2213, default 1): <Enter>
Using default value 1
Last cylinder or +size or +sizeM or +sizeK (1-2213, default 2213): <Enter>
Using default value 2213

Command (m for help): type
Selected partition 1
Hex code (type L to list codes): fd
Changed system type of partition 1 to fd (Linux raid autodetect)
```

```
Command (m for help): write
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

После записи меток разделов остальных двух дисков можно приступить к построению массива RAID. В большинстве современных дистрибутивов для управления массивами RAID используется единая команда `mdadm`. В предыдущих версиях дистрибутива RHEL использовался комплект инструментальных средств `raidtools`, но поскольку команда `mdadm` предоставляет больше возможностей и проще в использовании, чем `raidtools`, именно ее мы и используем в этом примере.

Следующая команда создает массив RAID 5 из трех разделов SCSI:

```
# mdadm --create /dev/md0 --level=5 --raid-devices=3 /dev/sdb1 /dev/sdc1
      /dev/sdd1
mdadm: array /dev/md0 started.
```

Файл `/proc/mdstat` отображает информацию о протекании процесса построения массива:

```
# cat /proc/mdstat
Personalities : [raid5]
md0 : active raid5 sdb1[3] sdc1[1] sdd1[2]
35566336 blocks level 5, 64k chunk, algorithm 2 [3/2] [_UU]
  [====>.................]  recovery = 22.4% (3999616/17783168) finish=5.1min
  speed=44800K/sec
unused devices: <none>
```

Этот файл всегда отражает текущее состояние системы RAID в структуре ядра. Особенно важно обращать внимание на этот файл после добавления нового или замены отказавшего диска.

По завершении сборки массива в файле `/var/log/messages` появляется соответствующее уведомляющее об этом сообщение:

```
RAID5 conf printout:
--- rd:3 wd:3 fd:0
disk 0, o:1, dev:sdb1
disk 1, o:1, dev:sdc1
disk 2, o:1, dev:sdd1
```

Команда создания массива выполняет также его “активизацию” (делает его доступным для использования), но при последующих перезагрузках активизацию массива придется выполнять отдельно — обычно из сценария начальной загрузки. Дистрибутивы RHEL, Fedora и SUSE содержат образцы сценариев начальной загрузки, выполняющие активирование массивов RAID.

Команда `mdadm` не нуждается в файле конфигурации, но будет его использовать, если тот указан (как правило — `/etc/mdadm.conf`). Мы настоятельно рекомендуем его использовать, поскольку он позволяет документировать конфигурацию массива RAID стандартным образом, и, соответственно, администраторам не приходится гадать, где искать необходимую информацию при возникновении проблем. Альтернатива использованию файла конфигурации — указание конфигурации в командной строке при каждом активировании массива.

Команда `mdadm --detail --scan` записывает текущие параметры настройки массива RAID в файл конфигурации. К сожалению, выводимая этой командой информация о

конфигурации недостаточно полна. Следующие команды создают полный файл конфигурации для рассматриваемого примера:

```
# echo DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1 > /etc/mdadm.conf
# mdadm --detail --scan >> /etc/mdadm.conf
# cat /etc/mdadm.conf
DEVICE /dev/sdb1 /dev/sdc1 /dev/sdd1
ARRAY /dev/md0 level=raid5 num-devices=3 UUID=21158de1:faaa0dfb:
      841d3b41:76e93a16
  devices=/dev/sdb1,/dev/sdc1,/dev/sdd1
```

Теперь команда `mdadm` может легко управлять массивом, считывая этот файл во время начальной загрузки или выключении системы. Чтобы активизировать массив во время начальной загрузки с помощью только что созданного файла `/etc/mdadm.conf` нужно выполнить команду

```
# mdadm -As /dev/md0
```

Для отключения массива вручную можно использовать команду

```
# mdadm -S /dev/md0
```

Итак, мы собрали три жестких диска в единый логический диск RAID. Теперь пора определить группы логических томов, в которых можно будет создавать динамически расширяемые (и сужаемые) файловые системы. Конфигурирование LVM состоит из нескольких отдельных этапов:

- создание (точнее говоря, определение) и инициализация физических томов;
- добавление физических томов в группу томов;
- создание логических томов в группе томов.

Комплект инструментов LVM2 выполняет все эти задачи и облегчает последующее управление томами. Чтобы ознакомиться с системой и предоставляемыми ею средствами, можно выполнить команду `man lvm`.

По терминологии LVM физические тома — это “объекты”, собранные в пулы хранилищ данных (“группы томов”). Однако этот термин несколько вводит в заблуждение, поскольку физические тома должны непосредственно соответствовать физическим устройствам. Такими устройствами могут быть диски, но ими могут быть и разделы диска или (как в данном примере) высокоуровневые объекты RAID, обладающие собственной базовой структурой.

Команды LVM начинаются с букв, которые указывают используемый уровень абстракции: команды `pv` манипулируют физическими томами, `vg` — группами томов, а `lv` — логическими томами.

В более ранних версиях LVM на начальном этапе нужно было запускать команду `vgscan`, но теперь это не требуется. Вместо этого начинать следует непосредственно с инициализации каждого физического устройства с помощью команды `pvcreate`. В данном примере мы используем только что созданное устройство `/dev/md0` типа RAID 5.

```
# pvcreate /dev/md0
Physical volume "/dev/md0" successfully created
```

Эта операция уничтожает все данные на диске или в разделе, поэтому при ее использовании необходимо быть чрезвычайно внимательными! Хотя в данном примере используется только одно физическое устройство, LVM позволяет добавлять в одну группу томов несколько устройств различных типов.

Теперь физическое устройство может быть добавлено в группу томов:

```
# vgcreate LVM1 /dev/md0
Volume group "LVM1" successfully created
```

Для получения информации о результатах своей работы можно использовать команду vgdisplay:

```
# vgdisplay LVM1
--- Volume group ---
VG Name               LVM1
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  1
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                0
Open LV               0
Max PV                0
Cur PV                1
Act PV                1
VG Size               33.92 GB
PE Size               4.00 MB
Total PE              8683
Alloc PE / Size       0 / 0
Free                  PE / Size8683 / 33.92 GB
VG UUID               nhkzzN-KHmY-BfV5-6F6Y-3LF8-dpd5-JM5lMp
```

В заключение необходимо создать логические тома в группе томов LVM1 и создать в них логические разделы. Оба логических раздела будут иметь одинаковый размер, равный 10 Гбайт:

```
# lvcreate -L 10G -n web1 LVM1
Logical volume "web1" created

# lvcreate -L 10G -n web2 LVM1
Logical volume "web2" created
```

Теперь, когда два логических тома web1 и web2 в группе томов LVM1 созданы, можно создать и монтировать файловые системы.

```
# mke2fs -j /dev/LVM1/web1
...
# mke2fs -j /dev/LVM1/web2
...
# mkdir /web1 /web2
# mount /dev/LVM1/web1 /web1
# mount /dev/LVM1/web2 /web2
```

Наконец файловые системы готовы к использованию. Остается только добавить новые файловые системы в файл /etc/fstab и выполнить перезагрузку системы, чтобы убедиться в нормальности их функционирования.

Действия по отношению к отказавшему диску

Тщательно созданная система выглядит весьма стройной, но в связи с множеством уровней, на которых она функционирует, все очень быстро может измениться в худшую сторону. В случае отказа жесткого диска или повреждения логического раздела (или просто при его переполнении) важно знать, как можно быстро исправить ситуацию. Для поддержания системы в работоспособном состоянии и устранения возникающих проблем служат те же средства, что и для описанного начального конфигурирования.

Рассмотрим случай отказа жесткого диска. Поскольку RAID 5 обеспечивает определенную избыточность данных, созданный нами массив RAID 5 будет успешно работать даже при отказе одного жесткого диска. Пользователи могут даже не заметить наличие проблем. Чтобы обнаружить проблему сразу после ее возникновения, следует постоянно следить за журнальными файлами системы (либо поручить выполнение этой задачи программе; для получения дополнительной информации обратитесь к разделу 10.4).

Команда `mdadm` предоставляет удобную опцию, которая имитирует отказ диска:

```
# mdadm /dev/md0 -f /dev/sdc1
mdadm: set /dev/sdc1 faulty in /dev/md0

# tail /var/log/messages
May 30 16:14:55 harp kernel: raid5: Disk failure on sdc, disabling device.
  Operation continuing on 2 devices
kernel: RAID5 conf printout:
kernel: --- rd:3 wd:2 fd:1
kernel: disk 0, o:1, dev:sdb1
kernel: disk 1, o:0, dev:sdc1
kernel: disk 2, o:1, dev:sdd1
kernel: RAID5 conf printout:
kernel: --- rd:3 wd:2 fd:1
kernel: disk 0, o:1, dev:sdb1
kernel: disk 2, o:1, dev:sdd1
```

Как видите, в системном журнале `/var/log/messages` информация об отказе (в данном случае имитированном) появляется сразу после его появления. Аналогичную информацию можно получить из файла состояния массива RAID, `/proc/mdstat`. На этом этапе администратор должен предпринять следующие действия:

- удалить диск из массива RAID;
- запланировать простой и выключение компьютера (при необходимости);
- заменить физический диск;
- добавить новый диск в массив.

Для удаления диска из конфигурации RAID необходимо использовать команду `mdadm`:

```
# mdadm /dev/md0 -r /dev/sdc1
mdadm: hot removed /dev/sdc1
```

Сразу после логического удаления диска его можно заменить. Дисководы, поддерживающие "горячую замену" позволяют выполнить эту операцию без выключения системы или ее перезагрузки.

Если компоненты RAID — неразбитые на разделы диски, их можно заменять только идентичными устройствами. Компоненты, представляющие собой логические разделы,

можно заменять разделами такого же размера, хотя с точки зрения соответствия пропускной способности лучше, чтобы аппаратные характеристики дисководов были идентичными. (Если конфигурация RAID построена поверх логических разделов, прежде чем добавлять заменяющий диск в массив, необходимо запустить утилиту `fdisk`, чтобы соответствующим образом определить логические разделы.)

В нашем примере отказ всего лишь имитирован, поэтому можно снова добавить диск в массив без замены какого-либо оборудования.

```
# mdadm /dev/md0 -a /dev/sdc1
mdadm: hot added /dev/sdc1
```

Как обычно, система Linux перестраивает массив и отражает протекание этого процесса в файле `/proc/mdstat`.

Перераспределение дискового пространства

Еще более распространенная ситуация, чем отказ дисков — заполнение логических разделов данными пользователей или журнальных файлов. Нам приходилось сталкиваться с самыми разнообразными ситуациями — от серверов, используемых в качестве личных хранилищ MP3-файлов, до серверов, заполненных ненужной перепиской подразделений.

Предположим, что раздел `/web1` увеличился больше, чем предполагалось, и нуждается в дополнительном дисковом пространстве. Для изменения размеров логических разделов LVM требуется всего несколько небольших шагов. Конкретные команды зависят от используемой файловой системы. Для файловой системы типа ext3 необходимо выполнить следующие действия:

- проанализировать текущую конфигурацию LVM;
- изменить размеры логических разделов с помощью команд `lvextend` и `ext2online`;
- проверить и подтвердить изменения.

К счастью, в группе томов было оставлено немного дополнительного места для увеличения раздела `/web1`, поэтому нам не нужно заимствовать дисковое пространство в другом томе. Для выяснения объема дискового пространства доступного в группе томов можно использовать команду `vgdisplay`, а для определения способа его перераспределения — команду `df`:

```
# vgdisplay LVM1
--- Volume group ---
VG Name               LVM1
System ID
Format                lvm2
Metadata Areas        1
Metadata Sequence No  3
VG Access             read/write
VG Status             resizable
MAX LV                0
Cur LV                2
Open LV               2
Max PV                0
Cur PV                1
Act PV                1
```

```
VG Size               33.92 GB
PE Size               4.00 MB
Total PE              8683
Alloc PE / Size       5120 / 20.00 GB
Free                  PE / Size3563 / 13.92 GB
VG UUID               nhkzzN-KHmY-BfV5-6F6Y-3LF8-dpd5-JM5lMp
```

```
# df -h /web1
Filesystem            Size      Used      Avail     Use%      Mounted on
/dev/mapper/LVM1-web1 9.9G      7.1G      2.3G      76%       /web1
```

Эти команды показывают, что объем свободного дискового пространства в группе томов составляет 13,92 Гбайт, а использование дискового пространства в разделе `/web1` — 76%. Добавим 10 Гбайт к разделу `/web1`.

Вначале необходимо использовать команду `lvextend`, чтобы добавить дисковое пространство в логический том, а затем применить команду `ext2online` для изменения размеров структур файловой системы так, чтобы они охватывали дополнительное дисковое пространство.

```
# lvextend -L+10G /dev/LVM1/web1
  Extending logical volume web1 to 20.00 GB
  Logical volume web1 successfully resized
# ext2online -d /dev/LVM1/web1
...
ext2_close
```

Вот и все! Чтобы увидеть выполненные изменения, можно снова просмотреть вывод команды `df`:

```
# df -h /web1
Filesystem            Size      Used      Avail     Use%      Mounted on
/dev/mapper/LVM1-web1 20G       7.1G      12G       38%       /web1
```

Пользователи ReiserFS перед выполнением команды `lvextend` должны выполнить размонтирование логического раздела. Кроме того, для изменения размера файловой системы вместо команды `ext2online` им придется использовать утилиту `resize_reiserfs`.

7.9. Монтирование дисков USB

Гибкие диски уходят в прошлое. Им на замену приходя удобные, быстрые и компактные диски USB. Существует множество разновидностей этих устройств: персональные "переносные" диски, цифровые камеры, проигрыватели iPod, большие внешние диски — вот лишь некоторые из них. Последние дистрибутивы Linux включают в себя поддержку этих удобных принадлежностей на уровне ядра.

При подключении диска USB вначале необходимо убедиться, что система Linux распознает это устройство. Следующая команда выводит список USB-устройств, обнаруженных ядром:

```
$ sudo /sbin/lsusb
Bus 001 Device 003: ID 0781:7108 SanDisk Corp.
Bus 001 Device 001: ID 0000:0000
```

В данном случае к системе подключен диск SanDisk. Если устройство подключено, но в списке оно отсутствует, возможно, ядро не поддерживает USB-устройства, и его нужно перекомпилировать.

Затем необходимо выяснить, как ядро идентифицирует диск, и какой файл устройства оно использует для его представления. Соответствующие сообщения ядра записываются в системный журнал.

```
$ sudo tail -n 20 /var/log/messages | grep kernel[8]
Jul 27 20:52:13 harp kernel: USB Mass Storage support registered.
Jul 27 21:02:57 harp kernel: usb 1-2: USB disconnect, address 2
Jul 27 21:14:09 harp kernel: ohci_hcd 0000:00:0f.2: wakeup
Jul 27 21:14:09 harp kernel: usb 1-2: new full speed USB device using addr 3
Jul 27 21:14:09 harp kernel: scsi3 : SCSI emulation for USB Storage devices
Jul 27 21:14:09 harp kernel: Vendor: SanDisk Model: Cruzer Titanium Rev: 2000
Jul 27 21:14:09 harp kernel: Type: Direct-Access ANSI SCSI revision: 02
Jul 27 21:14:09 harp kernel: SCSI device sde: 512-byte hdwr sectors (520 MB)
Jul 27 21:14:09 harp kernel: sde: Write Protect is off
Jul 27 21:14:09 harp kernel: sde: assuming drive cache: write through
Jul 27 21:14:10 harp kernel: sde: sde1
Jul 27 21:14:10 harp kernel: Attached scsi removable disk sde at scsi3, channel
  0, id 0, lun 0
```

Из сообщений ядра видно, что данное устройство — диск SanDisk Cruzer Titanium емкостью 520 Мбайт. Ядро связало с диском устройство `/dev/sde`, и диск содержит только один логический раздел `sde1`.

Прежде чем его можно будет использовать, диск должен быть смонтирован системой Linux. Поэтому нужно создать точку монтирования и смонтировать диск:

```
$ sudo mkdir /mnt/usb
$ sudo mount /dev/sde1 /mnt/usb
```

Теперь диск смонтирован в файле `/mnt/usb` и готов к использованию. Чтобы упростить процесс последующего использования диска, в файл `/etc/fstab` можно добавить следующую строку:

```
/dev/sde1      /mnt/usb    auto     users,noauto,uid=ben,gid=users   0   0
```

Использованные опции автоматически определяют тип файловой системы и предоставляют пользователю `ben` право монтирования и записи на диск.

7.10. Упражнения

7.1. Какие разъемные соединения используются в каждой из спецификаций SCSI? Помимо различий в разъемах, что еще нужно учитывать при работе со SCSI-устройствами разных типов?

7.2. В чем разница между форматированием диска и разбивкой его на разделы? В чем разница между разбивкой на разделы и созданием файловых систем?

7.3. Перечислите команды (вместе с аргументами), которые необходимы для создания файловой системы на жестком диске при следующих обстоятельствах:

а) диск будет служить для хранения домашних каталогов пользователей;

б) диск будет использоваться в качестве области подкачки;

[8] В системах Debian и Ubuntu сообщения записываются в файл `/var/log/kern.log`.

в) на диске будет храниться очередь почтовых сообщений организации, с огромным количеством спама;

г) на диске будет храниться база данных MySQL InnoDB.

7.4. Комплект инструментальных средств LVM предоставляет очень большие возможности, но при недостаточном его понимании может вызывать у пользователя чувство растерянности. Попрактикуйтесь в добавлении, удалении и изменении размеров дисков в группе томов. Покажите, как можно успешно удалить устройство из одной группы томов и добавить его в другую.

☆ 7.5. Поищите в литературе или в Internet тесты производительности жестких дисков и определите наиболее производительные SCSI- и IDE-диски. Насколько дороже обойдется покупка SCSI-диска, и какой выигрыш производительности будет получен в результате?

☆ 7.6. Подключите к системе новый диск. Создайте на диске раздел, являющийся копией корневого раздела. Инсталлируйте в него ядро и загрузитесь с диска. Ведите журнал команд, которые требуются для решения этой задачи (воспользуйтесь утилитой `script`). (Необходим доступ с правами суперпользователя.)

☆ 7.7. Что такое суперблок и для чего он нужен? Найдите определение структуры суперблока файловой системы ext3fs в файлах заголовков ядра и выясните, что представляет собой каждое поле этой структуры.

☆ 7.8. Используйте команду `mdadm` и ее опцию `-f` для имитации отказа диска в массиве RAID. Удалите диска из массива, а затем снова добавьте его. Как выглядит файл `/proc/md-stat` на каждом из этапов?

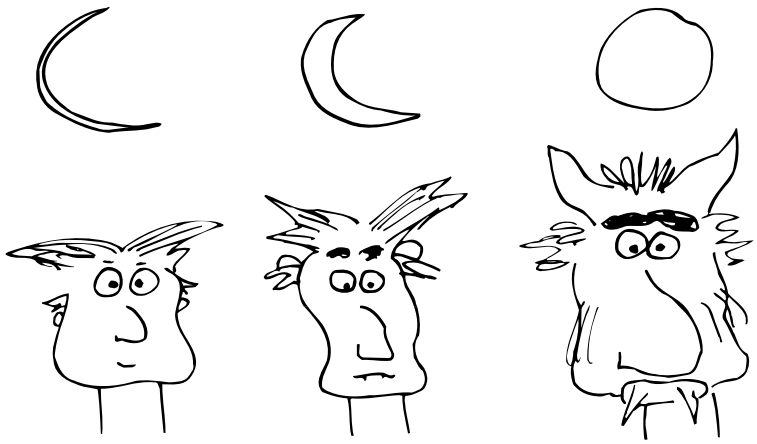
☆☆ 7.9. Какие информационные поля есть у индексного дескриптора в файловой системе ext3fs? Приведите содержимое индексного дескриптора файла `/etc/motd`. Где хранится имя этого файла? (Воспользуйтесь командами `hexdump` и `ls -i`.)

☆☆ 7.10. Просмотрите содержимое каталога (каталожного файла) с помощью утилиты `hexdump` или `od`. Каждая запись переменной длины представляет отдельный файл этого каталога. Найдите запись, соответствующую какому-либо реальному каталогу, и объясните назначение каждого из ее полей. Затем таким же способом проанализируйте каталог `lost+found` какой-нибудь файловой системы. Почему в каталожном файле так много имен, хотя сам каталог `lost+found` выглядит пустым?

☆☆☆☆☆ 7.11. Напишите программу, которая отображает содержимое файлов `/etc/motd` и `/etc/termcap`. Но не обращайтесь к файлам напрямую: откройте файл устройства, соответствующий корневому разделу, и воспользуйтесь системными вызовами `seek` и `read` для поиска нужных блоков данных. Файл `/etc/motd` обычно невелик и содержит только “информационные” блоки. При работе с файлом `/etc/termcap`, скорее всего, потребуется анализировать “ссылочные” блоки. Совет: найдя в системных файлах заголовков описание структуры индексного дескриптора, убедитесь, что это дескриптор, хранящийся на диске, а не в ядре. (Необходим доступ с правами суперпользователя.)

Глава 8

Периодические процессы



Ключом к сохранению постоянного контроля над системой является автоматизация максимально возможного числа задач. Например, сценарий `adduser` может подключать новых пользователей быстрее, чем это сделает администратор вручную, причем с гораздо меньшей вероятностью ошибки. Практически любую задачу можно запрограммировать в сценариях интерпретатора команд либо на языках Perl, Python или Expect.

В некоторых случаях желательно, чтобы сценарий или команда выполнялись без вмешательства оператора. Можно, в частности, сделать так, чтобы сценарий проверял (скажем, каждые полчаса), как работают сетевые маршрутизаторы и мосты, и при обнаружении проблем посылал администратору сообщение по электронной почте[1].

8.1. Демон `cron`: системный планировщик

В Linux периодическим выполнением программ управляет демон `cron`. Он запускается на этапе начальной загрузки системы и выполняется в фоновом режиме до тех пор, пока система не будет выключена. Демон `cron` читает один или несколько файлов конфигурации, содержащих списки командных строк и расписание их вызова. Командные строки обрабатываются интерпретатором `sh`, поэтому почти все, что можно делать в данном интерпретаторе вручную, разрешается перепоручать демону `cron`[2].

---

[1] Многие организации идут еще дальше: в них сценарий отправляет текстовое сообщение на пейджер администратора, уведомляя его о возникновении проблем.

[2] В Linux файл `/bin/sh` является ссылкой на интерпретатор `bash`. Можно настроить демон `cron` на использование других интерпретаторов.

Демон cron появился в UNIX еще в 70-х гг. В Linux включена более современная, улучшенная версия, известная как "Vixie-cron". Ее автором является Пол Викси (Paul Vixie).

Файлами конфигурации демона управляет команда crontab (сокращение от "cron table" — таблица демона cron), поэтому их называют crontab-файлами. Они ищутся в каталоге /var/spool/cron (или его подкаталогах tabs в SUSE либо crontabs в Debian) и /etc/cron.d. Просматривается также содержимое файла /etc/crontab.

Пользовательские crontab-файлы хранятся в каталоге /var/spool/cron. Для каждого пользователя создается не более одного такого файла. В качестве имени файла выбирается регистрационное имя пользователя, которому он принадлежит, и на основании этого имени демон cron выясняет, какое значение UID нужно использовать при выполнении команд, содержащихся в файле.

Таблицы расписаний, которые определяют график запуска системных административных заданий, хранятся в файле /etc/crontab и каталоге /etc/cron.d. Их формат слегка отличается от формата пользовательских crontab-файлов, поскольку содержащиеся в них команды могут выполняться от имени любого пользователя. Обработка записей файла /etc/crontab и каталога /etc/cron.d демоном cron выполняется одинаково. В целом файл /etc/crontab предназначен для того, чтобы системный администратор редактировал его вручную, тогда как в каталог /etc/cron.d программы могут записывать собственные таблицы расписаний.

При запуске демон cron читает все свои конфигурационные файлы, помещает их в память и переходит в режим ожидания. Раз в минуту демон "пробуждается", проверяет даты модификации crontab-файлов, загружает те файлы, которые изменились, а затем выполняет команды, запланированные на данную минуту, после чего снова "засыпает".

По непонятным причинам в системах Red Hat демон cron переименован в crond.

Выполненные им действия демон cron фиксирует в системном журнале, помечая записи меткой "cron". При этом большинство сообщений передается на информационном уровне ("info"). Как правило, конфигурации системного журнала, используемые по умолчанию, направляют данные журнала демона cron в отдельный файл.

8.2. Формат crontab-файлов

Все crontab-файлы в системе имеют одинаковый формат. Комментарии начинаются со знака решетки в первой позиции строки. Каждая строка, не являющаяся комментарием, содержит шесть или семь полей и представляет одну команду:

минута час день месяц день_недели [*имя_пользователя*] *команда*

Первые шесть полей отделяются друг от друга пробелами, но в поле *команда* пробел выполняет свою обычную функцию разделителя аргументов. Поле *имя_пользователя* встречается только в файле /etc/crontab, а также в файлах каталога /etc/cron.d. Оно определяет пользователя, от имени которого должна выполняться команда. Это поле не нужно в пользовательских crontab-файлах (тех, что находятся в каталоге /var/spool/cron), так как значение UID определяется на основании имени файла.

В полях *минута*, *час*, *день*, *месяц* и *день_недели* содержится информация о времени запуска команды. Описание этих полей приведено в табл. 8.1.

Таблица 8.1. Спецификации времени в crontab-файле

Поле	Описание	Диапазон
минута	Минута часа	от 0 до 59
час	Час дня	от 0 до 23
день	День месяца	от 1 до 31
месяц	месяц года	от 1 до 12
день_недели	День недели	от 0 до 6 (0 = воскресенье)

Каждое из полей времени может содержать:

- звездочку, которая обозначает любую цифру;
- целое число, трактуемое буквально;
- два разделенных дефисом целых числа, задающих диапазон значений;
- целые числа или диапазоны, разделенные запятыми (искомое время соответствует любому из указанных значений).

Диапазоны значений могут содержать значение шага поиска. Например, последовательность 0,3,6,9,12,15,18 можно записать короче: 0-18/3. Можно также использовать сокращенные названия месяцев и дней недели, но не в сочетании с диапазонами.

Например, строка

```
45  10  *  *  1-5
```

означает “10 часов 45 минут, с понедельника по пятницу”. Совет: никогда не ставьте звездочку в первое поле, иначе команда будет выполняться каждую минуту.

С полями *день_недели* и *день* сопряжена потенциальная двусмысленность, которую необходимо учитывать. День можно рассматривать как день недели и число месяца. Если указаны оба поля, то искомому дню достаточно удовлетворять одному из этих требований, чтобы пройти отбор. Например, спецификация

```
0,30  *  13  *  5
```

означает “каждые полчаса по пятницам и каждые полчаса тринадцатого числа месяца”, но не “каждые полчаса в пятницу, 13-го”.

Поле *команда* содержит командную строку, выполняемую интерпретатором sh. Это может быть любая допустимая команда интерпретатора (брать ее в кавычки не нужно). Считается, что поле *команда* продолжается до конца строки и может содержать пробелы и символы табуляции.

Знак процента (%) используется вместо символа новой строки в поле *команда*. В реальную команду включается только текст до первого знака процента, а остальные строки передаются команде в качестве стандартного входного потока.

Приведем несколько примеров допустимых команд crontab-файла:

```
echo The time is now `date` > /dev/console
mail -s Reminder evi@anchor % Don't forget to write your chapters.
cd /etc; /bin/mail -s "Password file" evi < passwd
```

А теперь рассмотрим полные примеры записей.

```
30  2  *  *  1      (cd /users/joe/project; make)
```

Эта строка будет активизироваться в 2:30 по понедельникам. Она инициирует выполнение утилиты make в каталоге /users/joe/project. Такую запись можно использовать для запуска длительного процесса компиляции в то время, когда в системе не работают другие пользователи. Вся выходная информация, выдаваемая командами crontab-файла, обычно посылается по электронной почте "владельцу" файла[3].

```
20   1   *   *   *   find /tmp -atime +3 -exec rm -f { } ';'
```

Эта строка будет активизироваться каждый день в 1:20. Соответствующая команда удаляет из каталога /tmp все файлы, к которым за последние 3 дня никто не обращался.

```
55   23   *  *   0-3,6 /staff/trent/bin/checkservers
```

Эта строка запускает сценарий checkservers в 23:55 каждый день, кроме четверга и пятницы.

В crontab-файлах допускается задавать значения переменных среды. Подробности можно узнать на man-странице crontab(5).

8.3. Управление crontab-файлами

Команда crontab *имя_файла* инсталлирует указанный crontab-файл, заменяя его предыдущую версию, если таковая имеется. Команда crontab -e проверяет копию crontab-файла текущего пользователя, загружает ее в текстовый редактор (указанный в переменной среды EDITOR) для последующего изменения, а затем повторно записывает файл в системный каталог. Команда crontab -l отображает содержимое crontab-файла, а команда crontab -r удаляет этот файл.

Пользователю root разрешается задавать аргумент *имя_пользователя*, чтобы можно было просматривать и редактировать crontab-файлы других пользователей. Например, команда crontab -u jsmith -r удаляет crontab-файл, принадлежащий пользователю jsmith.

Будучи вызванными без аргументов, большинство версий команды crontab будут пытаться прочитать crontab-файл из своего стандартного входного потока. Если этот режим был активизирован случайно, не пытайтесь выйти из него, нажимая <Ctrl+D>, так как весь crontab-файл будет удален. Необходимо нажать <Ctrl+C>. В некоторых версиях для того, чтобы команда crontab обратилась к своему стандартному входному потоку, в качестве аргумента *имя_файла* ей необходимо передать значение dash.

Два файла конфигурации /etc/cron.deny и /etc/cron.allow содержат информацию о том, кому из пользователей разрешено предоставлять crontab-файлы. Если файл cron.allow существует, то он содержит список пользователей, имеющих доступ к демону cron (по одному имени в строке). Пользователи, которые в списке отсутствуют, не имеют права выполнять команду crontab. Если файла cron.allow нет, проверяется файл cron.deny. Он также содержит список пользователей, но с противоположным назначением: доступ разрешен всем, кроме лиц, указанных в списке. Если нет ни одного из этих файлов, в большинстве систем только пользователь root имеет право создавать crontab-файлы. (В системах Debian и Ubuntu это право по умолчанию предоставлено всем пользователям.)

Необходимо подчеркнуть, что права доступа регулируются командой crontab, а не демоном cron. Если пользователь найдет способ обходным путем поместить в систем-

[3] Т.е. пользователю, именем которого назван файл. Фактическим владельцем crontab-файлов в общем случае является пользователь root.

ный каталог свой crontab-файл, демон cron будет благополучно выполнять указанные в нем команды.

8.4. Стандартные применения демона cron

Существует целый ряд стандартных задач, решать которые удобнее всего с помощью демона cron. Соответственно, большинство записей crontab-файла пользователя root служит именно этим целям. В данном разделе мы рассмотрим подобные задачи и элементы crontab-файла, обеспечивающие их реализацию.

Linux-системы часто поставляются с готовыми crontab-файлами, содержащимися преимущественно в каталоге /etc/cron.d. Если необходимо отключить стандартные записи, превратите их в комментарии, вставив в начало каждой строки знак решетки. Не удаляйте сами записи: возможно, они еще пригодятся.

Помимо файлов каталога /etc/cron.d в дистрибутивах Linux есть также crontab-файлы, организующие запуск сценариев в ряде заранее известных каталогов. Это позволяет программам добавлять собственные периодические задания, не редактируя crontab-файлы. Например, сценарии, помещаемые в каталог /etc/cron.daily, запускаются раз в день, а сценарии каталога /etc/cron.weekly — раз в неделю.

Многие организации сталкиваются с незначительными, но повторяющимися сетевыми проблемами из-за того, что администраторы сконфигурировали демон cron для выполнения одной и той же команды на сотнях компьютеров в одно и то же время. Синхронизация часов всех компьютеров посредством протокола NTP еще больше усугубляет проблему. Эту проблему легко устранить с помощью сценария выполнения команд со случайной задержкой или путем настройки файла конфигурации, однако она может с трудом поддаваться диагностированию, поскольку ее признаки быстро полностью исчезают.

Чистка файловой системы

В любой Linux-системе есть ненужные файлы (естественно, речь идет не о системных файлах). Например, при аварийном завершении программы ядро создает файл core, содержащий образ адресного пространства программы[4]. Такие файлы используются разработчиками программного обеспечения, но для администраторов это бесполезная трата дискового пространства. Пользователи часто вообще не знают о назначении файлов core, поэтому предпочитают не брать на себя ответственность за их удаление.

📖 NFS рассматривается в главе 16.

Еще один источник лишних файлов — это NFS (Network File System — сетевая файловая система). Серверы NFS сохраняют файлы, которые уничтожены в локальной системе, но продолжают использоваться удаленным компьютером. В большинстве реализаций таким файлам присваиваются имена вида .nfs*xxx*, где *xxx* — числовой код. Бывает, что об этих файлах забывают и оставляют на диске, думая, что они удалены.

Многие программы создают в каталоге /tmp или /var/tmp временные файлы, которые по той или иной причине не удаляются. Некоторые программы, особенно текстовые редакторы, делают резервные копии каждого файла, с которым они работают.

[4] Термин “core” (букв. “сердечник”) является синонимом слова “memory” (память). Этот термин родился в ранних вычислительных системах, где в качестве запоминающих элементов использовались маленькие ферритовые кольца, закрепленные на пересечениях проводников.

Частичное решение проблемы файлового "мусора" состоит в регулярном (каждую ночь) восстановлении свободного дискового пространства. В современных системах обычно имеются встроенные средства подобной очистки, поэтому необходимо убедиться в том, что они соответствуют потребностям организации.

Во всех показанных ниже примерах для удаления ненужных файлов применяется команда `find`.

```
find / -xdev -type f '(' -name core -o name 'core.[0-9]*' ')' -atime +7
  -exec rm -f { } ';'
```

Эта команда удаляет файлы с именем `core`, которые не использовались в течение недели. Аргумент `-xdev` гарантирует, что команда `find` будет выполнена только в пределах корневой файловой системы, что очень важно в сетях, где возможно монтирование множества файловых систем посредством NFS. Если необходимо очистить несколько файловых систем, задайте для каждой из них свою команду (помните, что каталог `/var` часто является отдельной файловой системой).

Системный администратор может определить соглашения по именованию `core`-файлов в файле `/proc/sys/kernel/core_pattern`. Вышеприведенная команда `find` выполняет обработку только имен, используемых по умолчанию (`core` или `core.`*`PID`*). Аргумент `-type f` имеет большое значение, поскольку исходный код ядра Linux содержит каталог с именем `core`, удаление которого весьма нежелательно.

```
find / -xdev -atime +3 '(' -name '#*' -o -name '.#*' -o -name '*.CKP' -o
  -name '*~' -o -name '.nfs*' ')' -exec rm -f { } ';'
```

Эта команда удаляет не использовавшиеся в течение трех дней файлы, имена которых начинаются с префикса `#`, `.#` или `.nfs` либо заканчиваются расширением `~` или `.CKP`. Это типичные признаки временных файлов и резервных файлов различных текстовых редакторов.

В целях повышения производительности некоторые администраторы используют опцию `noatime` команды `mount`, чтобы предотвратить хранения файловой системой временных меток момента обращения к файлам. Подобная конфигурация помешает работе обеих вышеприведенных команд `find`, поскольку файлы будут казаться такими, к каким не было обращений, даже если они недавно были активными. К сожалению, в случае ошибки команда удаляет файлы. Поэтому, прежде чем использовать приведенные команды, убедитесь, что система поддерживает временные метки доступа к файлам.

```
cd /tmp; find . ! -name . ! -name lost+found -type d -mtime +3
  -exec /bin/rm -rf { } ';'
```

Эта команда рекурсивно удаляет все подкаталоги каталога `/tmp`, не модифицировавшиеся в течение трех дней. Обычные файлы в каталоге `/tmp` удаляются сценариями запуска системы, но в некоторых системах подкаталоги не удаляются. Если существует каталог `lost+found`, то он не включается в число удаляемых. Это важно, если каталог `/tmp` является отдельной файловой системой. Подробнее о каталоге `lost+found` рассказывалось в разделе 7.6.

Если в системе применяется одна из вышеперечисленных команд, следует уведомить пользователей о том, какими принципами руководствуется администратор при чистке файловой системы.

Распространение конфигурационных файлов по сети

📖 Подробнее о распространении конфигурационных файлов по сети рассказывается в главе 17.

При сетевом администрировании во многих случаях удобно пользоваться единой версией конфигурационных файлов, например базой почтовых псевдонимов (обычно хранится в файле `/etc/mail/aliases`). Основные версии этих файлов можно распространять по сети каждую ночь с помощью утилиты `rsync`, `rdist` или сценария на языке Expect.

Иногда необходима последующая обработка файлов. Например, если в файле `sendmail.cf` не задана опция `AutoRebuildAliases`, может потребоваться, чтобы пользователь выполнял команду `newaliases` для преобразования текстового файла почтовых псевдонимов в хешированный формат, используемый программой `sendmail`. Может понадобиться и загрузка файлов в административную базу данных, например в NIS.

Ротация журнальных файлов

Linux хорошо управляет большинством своих журнальных файлов, но некоторые файлы увеличиваются бесконечно до тех пор, пока пользователь вручную не очистит их. Существуют разные способы решения этой проблемы. Самый простой из них — периодически усекать файлы.

Более консервативный подход заключается в хранении нескольких старых версий каждого журнального файла и циклическом их использовании по-новому. Это называется ротацией. Ротация предотвращает бесконтрольное разрастание журнальных файлов и в то же время позволяет сохранять информацию о последних изменениях. Такую процедуру лучше всего планировать с помощью демона `cron`. Подробнее о ротации рассказывается в разделе 10.1.

8.5. Другие планировщики: демоны anacron и fcron

В общем случае демон `cron` не возвращается к тем командам, время запуска которых было пропущено из-за того, что система была отключена или произошло внезапное изменение системного времени. Однако демон `Vixie-cron` позволяет правильно выполнять задачи, выполняемые реже одного раза в час, при изменениях системного времени менее трех часов. Этот демон успешно справляется с такими небольшими изменениями, как переход на летнее/зимнее время.

Портативные и другие компьютеры, включение которых происходит бессистемно, в большей степени зависят от работы демона `cron` и наличие резервных копий предоставляет им определенные преимущества. Демон `anacron`, представляющий собой планировщик, ориентированный на временные интервалы, может служить прекрасным дополнением демону `cron`. Если он не включен в стандартный инсталляционный пакет, его можно установить с помощью системы управления стандартным пакетом.

В среде `anacron` вместо того, чтобы планировать выполнение конкретной команды в 2:00 в ночь на понедельник, можно потребовать ее выполнения еженедельно. В отличие от `cron`, демон `anacron` записывает время последнего выполнения каждой команды и поэтому для определения того, нужно ли ее выполнять снова, он может просто сравнивать это значение, указанный временной интервал и текущее время.

`anacron` предоставляет несколько функций, которые помогают уберечь только что загруженный компьютер от “захлестывания” волной запланированных на одно время задач обслуживания. Каждая задача может иметь собственную задержку запуска, а демон `anacron` позволяет помещать выполняемые команды в очередь, чтобы одновременно активной была только одна из них.

С точки зрения выполнения административных задач планирование с привязкой к временным интервалам более целесообразно, чем планирование выполнения в фиксированные моменты времени. К сожалению, демон `anacron` требует, чтобы интервалы были выражены в днях, и это существенно ограничивает его применение в качестве основного планировщика.

Сам демон `anacron` должен запускаться из `cron`. Хотя интервалы `anacron` устанавливаются в днях, имеет смысл запускать его чаще, чем раз в день — если бы демон `cron` мог постоянно запускать `anacron` ежедневно в одно и то же время, вероятно, без последнего можно было бы обойтись. Аналогично, `anacron` целесообразно запускать при начальной загрузке системы.

Более активную разновидность демона `cron`, которая включает `anacron`-подобные функции, представляет демон `fcron` (см. `fcron.free.fr`). В отличие от `anacron`, демон `fcron` разработан в качестве явной замены демона `Vixie-cron`. Он допускает использование ряда более сложных спецификаций времени, которые не могут быть выражены в языках конфигураций демонов `cron` и `anacron`. Тем не менее, все же стоит подумать, оправдывают ли эти функции усложнение демона `fcron` и проблемы, связанные с использованием нестандартного планировщика.

8.6. Упражнения

8.1. Локальный пользователь системы стал злоупотреблять своим правом создания `crontab`-файлов, периодически запуская ресурсоемкие задания. Вы несколько раз безуспешно просили его прекратить это делать и теперь вынуждены лишить его соответствующих привилегий. Перечислите действия, которые необходимо предпринять для того, чтобы удалить текущий `crontab`-файл пользователя и запретить ему впредь создавать такие файлы.

8.2. Придумайте три задания, которые требуется запускать регулярно (те, что были приведены в главе, — не в счет). Составьте `crontab`-записи для каждого задания и укажите, в какие файлы их необходимо поместить.

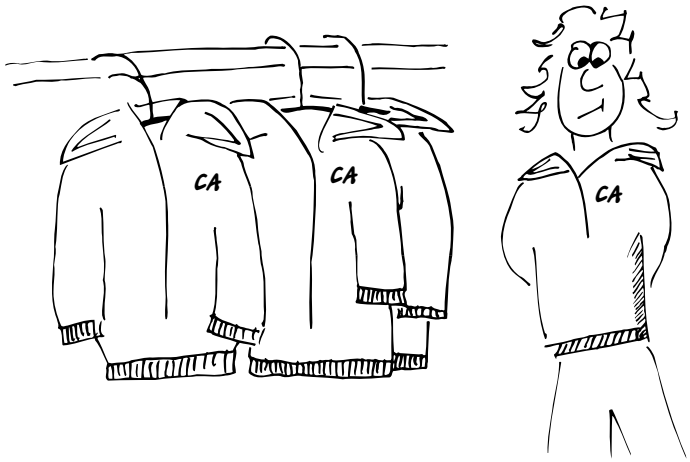
8.3. Найдите `crontab`-файлы в своей системе и выберите из них три записи. Опишите, когда активизируется каждая из них, что она делает и зачем, по вашему мнению, она нужна. (Необходим доступ с правами суперпользователя.)

☆ 8.4. Напишите сценарий, который синхронизирует ваши конфигурационные файлы (`~/.[a-z]*`) между всеми компьютерами, на которых у вас есть учетная запись. Сделайте так, чтобы этот сценарий регулярно запускался демоном `cron`. (Можно ли просто копировать все файлы, имена которых начинаются с точки? Нужно ли создавать резервные копии файлов, заменяемых на целевом компьютере, перед их перезаписью?)

☆ 8.5. Используя `man`-страницы команд `du`, `sort` и `head` в качестве справочного руководства, напишите сценарий, который определяет, какие десять домашних каталогов пользователей являются самыми крупными в системе. Запланируйте выполнение этого сценария каждый понедельник ровно в полночь и сделайте так, чтобы сценарий посылал вам результаты своей работы по электронной почте. Совет: используйте сортировку в обратном порядке числовых значений. (Необходим доступ с правами суперпользователя.)

Глава 9

Резервное копирование



В большинстве коммерческих организаций информация, хранящаяся в компьютерном виде, стоит дороже самих компьютеров. Кроме того, ее гораздо труднее восстановить. Защита этой информации является одной из наиболее важных (и, к сожалению, самых трудоемких) задач системного администратора.

Потерять важную информацию можно как угодно. К порче файлов данных приводят ошибки в программном обеспечении. Пользователи случайно удаляют то, над чем работали всю жизнь. Хакеры и обозленные служащие стирают жесткие диски. Сбои в аппаратных средствах и стихийные бедствия выводят из строя целые машинные залы.

Резервные копии позволяют администратору восстанавливать файловую систему (или любую ее часть) в том состоянии, в каком она находилась на момент последнего копирования. Резервное копирование должно осуществляться тщательно и строго по графику. Кроме того, устройства резервного копирования и сами носители должны регулярно проверяться на предмет корректной работы.

Безопасность резервных носителей является важным элементом корпоративной стратегии. Ответственные руководители должны четко понимать, какой цели служат архивы данных и как они должны эксплуатироваться. Для университетской компьютерной лаборатории потерять результаты работы за один день не так страшно, как для торговой компании, регулярно принимающей заказы от десятков или сотен клиентов.

Мы начнем эту главу с описания общих принципов резервного копирования, после чего ознакомимся с наиболее распространенными устройствами создания резервных копий и их носителями (их преимуществами, недостатками, ценами на них). Затем будут рассмотрены стандартные команды резервного копирования и архивирования в Linux и

даны пояснения, в каких ситуациях предпочтительнее та или иная команда. Далее речь пойдет о планировании процедуры резервного копирования и об особенностях работы популярных утилит `dump` и `restore`.

В завершение главы читатели познакомятся с бесплатным пакетом сетевого резервного копирования Bacula и несколькими его коммерческими альтернативами.

9.1. Принципы резервного копирования

Прежде чем приступить к подробному рассмотрению резервного копирования, мы хотели бы поделиться с читателями опытом, который приобрели за долгие годы практической работы (иногда эти знания давались очень тяжело). Ни один из предлагаемых советов не является абсолютной истиной, но чем чаще вы будете им следовать, тем спокойнее будет протекать процесс резервного копирования.

Создавайте резервные копии на одном компьютере

Многие специализированные утилиты позволяют создавать резервные копии по сети. Производительность процесса несколько ухудшается, но это компенсируется легкостью администрирования. Мы убедились, что лучше всего выполнять на центральном компьютере сценарий, который запускает утилиту `rdump` (посредством интерпретатора `rsh` или `ssh`) на всех компьютерах, где необходимо провести резервное копирование. Можно также воспользоваться программным пакетом (коммерческим, бесплатным или условно бесплатным), автоматизирующим этот процесс. Все архивы должны помещаться на одно устройство резервного копирования (в режиме без перемотки, естественно).

Если размер системы слишком велик и одного носителя недостаточно, все равно следует пытаться сделать систему резервного копирования как можно более централизованной. Это упростит администрирование и позволит проверять целостность резервных копий. Часто на сервере можно установить несколько устройств резервного копирования, и это не повлияет на его производительность.

Маркируйте носители

Маркируйте носители с резервными копиями системы максимально разборчиво и полно — неподписанная лента по существу бесполезна. Маркируйте каждый носитель уникальным образом в соответствии с его содержимым. На коробку следует нанести подробную информацию (например, список файловых систем и даты создания архивов).

Следует стремиться к тому, чтобы корневую файловую систему важного сервера и файловые системы `/usr` можно было восстановить, не заглядывая в сценарии архивирования. Резервные носители для этих файловых систем необходимо маркировать с указанием формата, точного синтаксиса команд, с помощью которых они были созданы, и других сведений, необходимых для восстановления архива без обращения к документации.

Существуют бесплатные и коммерческие программы создания наклеек для различных носителей. Советуем приобрести одну из них, дабы избавиться от ненужной головной боли. Например, изготовители наклеек, предназначенных для заполнения на лазерных принтерах, обычно поставляют с ними и программное обеспечение (для Windows) генерирования наклеек. Еще лучше приобрести специализированный принтер наклеек для Windows. Такие принтеры не дороги и прекрасно работают.

Правильно выбирайте периодичность резервного копирования

Чем чаще выполняется резервное копирование, тем меньше данных будет потеряно в случае сбоя системы. Процесс резервного копирования, однако, достаточно трудоемок и ресурсоемок. Системный администратор должен обеспечить приемлемую целостность данных при разумном уровне затрат времени и средств.

В интенсивно эксплуатируемых системах наиболее удобный вариант — создавать резервные копии файловых систем с домашними каталогами пользователей каждый рабочий день. В системах, которые используются менее интенсивно или в которых данные изменяются не столь часто, можно делать копии несколько раз в неделю. В маленькой системе с одним пользователем достаточно делать резервную копию раз в неделю. В целом периодичность определяется объемом данных, которые могут быть утеряны с момента последнего создания резервной копии.

Будьте осмотрительны при выборе архивируемых файловых систем

Файловые системы, которые изменяются редко, не нужно архивировать столь же часто, как домашние каталоги пользователей. Если в файловой системе изменяется лишь несколько файлов (например, файл `/etc/passwd` в корневой системе), их можно каждый день копировать в другой раздел, резервные копии которого создаются регулярно.

Если каталог `/tmp` является отдельной файловой системой, то архивировать его не нужно. Он не содержит ничего существенного, поэтому нет оснований сохранять его. Кому-то это может показаться очевидным, но нам известна крупная фирма, в которой ежедневно создается резервная копия этого каталога.

Старайтесь умещать каждодневные архивы на одном носителе

📖 Демон `cron` подробно описан в главе 8.

Было бы прекрасно иметь возможность создавать ежедневные резервные копии всех пользовательских файловых систем на одной ленте. Впрочем, при наличии устройств большой емкости, таких как накопители DLT, AIT и LTO, это вполне реально, но не всякая организация решится приобрести столь дорогие устройства. Процесс резервного копирования можно автоматизировать, ежедневно монтируя резервный носитель перед завершением работы и поручая выполнение резервного копирования демону `cron`. В этом случае запись резервных копий будет производиться тогда, когда файлы не будут претерпевать изменений, а сам процесс будет оказывать минимальное влияние на работу пользователей.

По мере того как привычки пользователей изменяются, а удаленный обмен данными становится все более популярным, “подходящий” временной интервал резервного копирования сокращается. Судя по нашему опыту, наилучшее время для резервного копирования — между 3:00 и 6:00 утра. Однако 80 Тбайт данных невозможно скопировать за три часа.

Еще одна существенная проблема — быстрое увеличение дискового пространства, связанное со снижением цен на жесткие диски. Теперь вряд ли удастся найти настоль-

ный компьютер фабричной сборки, имеющий меньше 100 Гбайт дискового пространства. Зачем очищать диски и устанавливать квоты, когда с помощью небольшого капиталовложения можно безболезненно решить проблему? К сожалению, слишком часто объем сетевых хранилищ данных делает полное резервирование невозможным.

Если ежедневные архивы невозможно разместить на одной ленте, есть несколько решений:

- купите ленточное устройство более высокой емкости;
- купите автозагрузчик или ленточную библиотеку, чтобы можно было вставлять несколько носителей в одно устройство;
- измените периодичность создания резервных копий;
- составьте более сложный сценарий архивирования;
- используйте несколько устройств резервного копирования.

Система автоматического резервного копирования должна регистрировать имена всех копируемых файловых систем. Это позволит при восстановлении быстро находить нужную файловую систему. Рекомендуем также записывать на коробке или на этикетке ленты порядок следования файловых систем. (Об этом уже говорилось раньше, но лучше повторить: для носителей, на которые записывается несколько архивов, необходимо использовать устройство, работающее в режиме без перемотки.)

Создавайте файловые системы, объем которых меньше емкости резервного носителя

Команда `dump` прекрасно справляется с копированием файловых систем на несколько носителей. Но в этом случае процесс копирования должен протекать в присутствии оператора или робота, меняющего носители, а сами носители необходимо аккуратно маркировать, чтобы при восстановлении данных не возникало никаких затруднений. Если нет реальных причин создавать большую файловую систему, не делайте этого.

Храните носители вне рабочего помещения

В большинстве организаций резервные копии хранят вне рабочего помещения во избежание их порчи вместе с основными данными в случае пожара или стихийного бедствия. Фраза “вне рабочего помещения” означает как сейф в банке, так и полку в доме у президента компании. Фирмы, специализирующиеся на безопасном хранении резервных носителей, обеспечивают для них надлежащий температурный режим и гарантируют их сохранность. Всегда следует выбирать фирму, пользующуюся хорошей репутацией. Существует множество фирм, специализирующихся на хранении данных, которые обеспечивают передачу данных по сети, но хранят их в надежном месте.

Выбор внешнего хранилища определяется тем, как часто нужно восстанавливать файлы и какие потери времени при этом допустимы. В некоторых организациях создают по две резервные копии в день на разных носителях: одна остается в рабочем помещении, а вторую сразу же выносят[1].

[1] Крупная финансовая компания, размещавшаяся во Всемирном торговом центре, хранила свои “внешние” резервные копии одним или двумя этажами ниже своих офисных помещений. Когда здание подверглось первой же террористической атаке, резервные ленты (как и сами компьютеры) были уничтожены. Поэтому “внешнее” хранилище должно быть действительно внешним.

Защищайте резервные копии

Дэн Гир (Dan Geer), специалист по вопросам безопасности, однажды сказал: “Что делает резервная копия? Гарантированно нарушает права доступа к файлам на расстоянии”. Ни больше, ни меньше!

Защищайте свои резервные носители. Они содержат все данные, имеющиеся в системе, и могут быть прочитаны всяким, кто имеет физический доступ к ленте. Носители нужно хранить не просто вне помещения, но также под замком, ключ от которого надежно спрятан. Если соответствующие услуги предоставляет коммерческая компания, она должна гарантировать конфиденциальность лент.

Еще одна возможность, о которой следует подумать — шифрование резервного носителя. Многие коммерческие утилиты резервного копирования делают шифрование сравнительно простым процессом. Вместе с тем, следует обеспечить сохранность и целостность ключей шифрования, а также их доступность при возникновении аварийных ситуаций.

Некоторые компании настолько заботятся о сохранности резервных копий, что даже создают их дубликаты.

Активность файловой системы во время создания архива должна быть низкой

Снятие резервных копий лучше проводить в периоды низкой активности системы, потому что изменения файлов могут привести к ошибкам в работе команды `dump`. Архивы следует создавать, когда в системе работает мало пользователей (ночью или в выходные дни), или же в определенное время доступ к системе следует предоставлять только утилите резервного копирования. Впрочем, это хорошо звучит в теории, но на практике реализуется редко. Пользователям зачастую нужен постоянный и круглосуточный доступ к системе. В наши дни практически невозможно выполнить резервное копирование при полном отсутствии каких-либо дисковых операций.

О специализированных файловых серверах рассказывается в разделе 16.5.

Существуют специализированные файловые серверы (например, те, что выпускаются компанией Network Appliance), которые позволяют в оперативном режиме создавать мгновенные копии файловой системы через указанные промежутки времени. Это дает возможность безопасно получать резервные копии работающей системы и является веским доводом в пользу специализированных файловых серверов.

Проверяйте состояние своих носителей

Известно немало ужасных историй о системных администраторах, которые не замечали проблем, связанных с созданием архивов, до тех пор, пока в системе не происходил серьезный сбой. Важными задачами администратора являются непрерывный контроль над соблюдением установленного порядка резервного копирования и проверка качества его выполнения. Ошибка со стороны оператора приводит к повреждению большего числа архивов, чем любая другая ошибка.

На первом этапе проверки следует заставить программу резервного копирования по окончании работы повторно прочесть все ленты[2]. Неплохо также просмотреть ленты на предмет наличия ожидаемого количества файлов. В идеале должна быть проконтролирована каждая лента, но это едва ли практически возможно в крупной организации, где каждый день задействуются сотни лент. Благоразумнее в подобной ситуации осуществлять выборочную проверку.

Более подробно команда `restore` рассматривается в разделе 9.4.

Часто имеет смысл создать таблицу содержимого каждой файловой системы (для этого можно использовать команду `restore -t` утилиты `dump`) и сохранить результат на диске. Полученные списки нужно называть так, чтобы их имена отражали связь с соответствующими лентами, например `okra:usr.Jan.13`. Недельный набор таких списков помогает быстро определить, на каком носителе находится пропавший файл. Для этого достаточно выполнить команду `grep` с именем файла в качестве аргумента и выбрать самый последний его экземпляр.

Успешное чтение таблицы содержимого свидетельствует о том, что архив создан нормально и что в случае необходимости наверняка можно будет восстановить файлы с носителя. Пробное восстановление файла, выбранного случайным образом, послужит еще более надежным доказательством возможности восстановления данных с этого носителя[3].

Периодически следует выполнять восстановление с произвольно выбранного резервного носителя, чтобы убедиться в том, что оно по-прежнему возможно. Иногда стоит читать файлы со старых резервных лент (месячной и даже годичной давности), ведь у накопителей временами нарушается центровка, и они утрачивают способность читать ими же записанные ленты. Некоторые компании выполняют восстановление старых носителей, но их услуги стоят достаточно дорого.

Еще один вид проверки — попробовать восстановить данные с носителя в другой аппаратной среде. Если машинный зал сгорит, пользы от осознания того, что резервную копию можно было бы прочитать на расплавившемся накопителе, будет мало. В прошлом цифровые аудиокассеты особенно часто страдали от этой проблемы, но теперь технология усовершенствована и носители стали значительно более надежными.

Определите жизненный цикл носителя

Срок службы всех носителей ограничен. Их можно и нужно использовать повторно, но не забывайте о предельном сроке эксплуатации, определяемом изготовителем носителя. Как правило, этот срок задается в виде количества проходов, которые способна выдержать лента. Каждая такая операция, как архивирование, восстановление и команда `mt fsf` (перемотка на один файл вперед), представляет собой один проход. Не ленточные носители обладают значительно более длительным сроком службы, который иногда выражают средним временем до отказа (mean-time-to-failure — MTTF).

Но прежде чем выбрасывать старые ленты, не забудьте стереть с них всю информацию или же сделать ее недоступной для считывания. В этом может помочь устройство для размагничивания магнитной ленты (большой электромагнит), но его следует держать вдали от компьютеров и активных носителей. Разрезание или вытягивание лен-

[2] Можно воспользоваться командой `restore -C`, чтобы сравнить содержимое ленты с исходным деревом каталогов.

[3] Команда `restore -t` читает только таблицу содержимого архива, хранящуюся в начале ленты. При реальном восстановлении конкретного файла проверяется более крупная область носителя.

ты из кассеты — не слишком надежная защита данных, поскольку ленту сравнительно легко склеить или намотать заново. Компании, специализирующиеся на уничтожении документов, за сравнительно невысокую плату предоставляют услуги по уничтожению магнитных лент.

Компонуйте данные с учетом резервного копирования

Когда есть столь дешевые и надежные дисковые устройства, возникает соблазн отказаться от архивирования всех имеющихся данных, сосредоточив усилия лишь на критически важных файлах, тем более что многие системы заполняются практически бесконтрольно. Но если разумно распланировать схему дискового хранения, то можно существенно упростить процесс резервного копирования. Начать необходимо с оценки потребностей.

- С данными каких типов придется иметь дело?
- Как часто будут изменяться данные различных типов?
- Насколько часто нужно выполнять резервное копирование, чтобы возможные потери причинили минимальный ущерб?
- Какие административные ограничения должны быть определены для данных?

Используя эту информацию, следует продумать схему хранения данных с учетом резервного копирования и возможного расширения системы в будущем. Лучше размещать каталоги проектов и домашние каталоги пользователей на выделенных файловых серверах, что упростит управление данными и позволит гарантировать их безопасность.

Будьте готовы к худшему

После разработки схемы резервного копирования изучите самый худший сценарий: система полностью уничтожена. Определите, сколько информации будет потеряно и сколько времени уйдет на восстановление системы (включая время на закупку нового оборудования). Затем посмотрите, удовлетворяют ли вас полученные результаты.

9.2. Устройства и носители, используемые для резервного копирования

Поскольку многие неисправности могут приводить к одновременному выходу из строя сразу нескольких аппаратных устройств, резервные копии следует помещать на съемные носители. Например, копирование содержимого одного жесткого диска на другой, конечно, лучше, чем ничего, но оно обеспечивает весьма незначительный уровень защиты на случай отказа контроллера. Хотя компании, организующие резервное копирование через Internet, становятся все более популярными, все же большинство резервных копий хранится локально.

Во многих типах носителей запись данных осуществляется с применением эффекта намагничивания. Такие носители подвержены разрушительному воздействию электрических и магнитных полей. Источниками магнитных полей являются звуковые динамики, трансформаторы и источники электропитания, неэкранированные электродвигатели, вентиляторы дисководов, электронно-лучевые мониторы. Даже длительное воздействие фонового излучения Земли может привести к повреждению магнитных носителей дан-

ных. По прошествии нескольких лет все ленты становятся нечитаемыми. Большинство ленточных носителей выдерживает три года, и если данные необходимо хранить дольше, то нужно либо использовать другие типы носителей, либо перезаписывать данные.

В следующих подразделах описываются различные виды носителей для хранения резервных копий. Они перечислены в порядке возрастания емкости.

Производители устройств резервного копирования любят указывать емкость, отталкиваясь от объема сжатых данных, оптимистично предполагая коэффициент сжатия 2:1 или выше. Ниже мы будем приводить реальное число байтов, которое физически можно записать на тот или иной носитель без учета сжатия.

Коэффициент сжатия влияет также на пропускную способность устройства. Если накопитель позволяет записывать данные на ленту со скоростью 1 Мбайт/с, а производитель заявляет о коэффициенте 2:1, то пропускная способность волшебным образом возрастает до 2 Мбайт/с. Как и в случае емкости, мы проигнорируем рекламную скорость записи и будем учитывать реальные показатели.

Цена и емкость — важнейшие характеристики носителя, но нельзя не учитывать и пропускную способность, которую он поддерживает. С быстродействующими устройствами удобнее работать, и они позволяют составлять более эффективный график резервного копирования.

В предыдущем издании этой книги в качестве возможных носителей резервного копирования мы упоминали гибкие диски, как обычные, так и особо высокой емкости (например, диски Zip). В прошлом они использовались повсеместно, но поскольку в настоящее время вряд ли удастся приобрести портативный компьютер, оборудованный дисководом гибких дисков, мы решили исключить этот тип носителей из числа рекомендуемых для создания резервных копий. Хотя гибкие диски удобны для обмена данными, относительно высокая стоимость носителя делает непрактичным их применение для резервного копирования.

Оптические носители: CD-R/RW, DVD±R/RW и DVD-RAM

При цене около $0,4 за штуку, диски CD и DVD хорошо подходят для резервного копирования небольших, изолированных систем. Емкость компакт-дисков CD составляет около 650 Мбайт, а дисков DVD — около 4,7 Гбайт. Емкость двухслойных DVD, которые лишь начинают завоевывать популярность, доходит до 8,5 Гбайт.

Дисководы с возможностью записи на эти носители выпускаются для всех стандартных шин (SCSI, IDE, USB и др.) и во многих случаях настолько дешевы, что по существу на их стоимость можно не обращать внимание. Мы уже привыкли к повседневному использованию дисков CD, но, судя по всему, время наибольшего расцвета этой технологии на исходе. Теперь, когда цены CD и DVD практически сравнялись, нет никаких причин использовать CD, а не DVD (если конечно не учитывать, что устройства чтения CD все еще более распространены).

Запись на оптические носители осуществляется фотохимическим способом посредством лазерного луча. Хотя достоверных данных нет, считается, что оптические носители значительно долговечнее, чем магнитные. Однако даже диски с однократной записью (CD-R, DVD-R и DVD+R) уступают по долговечности фабричным (штампованным) компакт-дискам CD и DVD.

Носители CD-R не подходят для регулярного создания резервных копий, но на них можно записывать архивы, которые потребуются в далеком будущем. Компакт-диски

CD-RW хорошо подходит для регулярного архивирования небольших объемов персональных данных.

Современные записывающие приводы DVD не уступают — а некоторые и превосходят — по скорости ленточным накопителям. Версии DVD-R (разработанная компанией Pioneer в 1997 г.) и DVD+R (разработанная в 2002 г.) предназначены для одноразовой записи. Носители DVD-RW, DVD+RW и DVD-RAM допускают многократную запись. Система DVD-RAM имеет встроенные средства обработки дефектов и поэтому более надежна, чем другие типы носителей. В то же время, эти носители значительно дороже остальных.

По прогнозам изготовителей потенциальный срок службы этих носителей при правильном их хранении должен составлять сотни лет. Изготовители рекомендуют хранить диски в отдельных коробках при постоянной температуре в диапазоне 41°–68°F (5°–20°C) и относительной влажности 30%–50% в защищенном от прямых солнечных лучей месте. Для маркировки дисков следует использовать только водорастворимые маркеры. Вероятно, в общем случае более реальным можно считать срок службы равный 1–5 годам.

Как показывают многочисленные исследования, выполненные независимыми экспертами, надежность оптических носителей в основном определяется фирмой-изготовителем. Это как раз тот случай, когда приобретение первоклассных носителей полностью окупается. К сожалению, даже изделия одного и того же изготовителя различаются по качеству. Поэтому ни один из них не в состоянии гарантировать стопроцентную надежность. В течение нескольких последних лет первые места по надежности занимали носители фирмы Taiyo Yuden, но это справедливо по отношению только тех дисков, которые были изготовлены в Японии. Носители этой же фирмы, изготовленные в других странах, обеспечивают менее надежную запись.

Недавно на рынке оптических носителей появились диски типа Blu-Ray, различные версии которых позволяют хранить 25–100 Гбайт данных. Их высокая емкость обеспечивается более короткой длиной волны (405 нм) лазера, используемого для чтения и записи (отсюда и название этого типа носителей “blue” — голубой). Первые проигрыватели этого типа появились в США в июне 2006 г. Судя по всему, эта технология заслуживает внимания и обещает стать хорошим весьма подходящим решением для резервного копирования.

Съемные жесткие диски (USB и FireWire)

Внешние накопители, подключаемые через порты USB или FireWire (IEEE1394), становятся все более распространенными. Обычно в качестве основного устройства хранения используется тот или иной жесткий диск, но флэш-память (вездесущие “портативные диски”) находит все большее распространение. Емкости этих устройств варьируются от менее 1 Мбайт до 600 Гбайт и более. В настоящее время максимальная емкость устройств флэш-памяти, встречающихся на рынке, составляет около 4 Гбайт.

Основным ограничением этих дисководов является скорость работы шины, но с появлением USB 2.0 и FireWire 800 обе разновидности обеспечивают вполне приличную пропускную способность в диапазоне 50-90 Мбайт/с.

В основном, долговечность устройств флэш-памяти определяется числом циклов записи. Предположительно устройства среднего класса рассчитаны на несколько миллионов циклов.

Малые лентопротяжные устройства: 8-миллиметровые и DDS/DAT

Различные варианты 8-миллиметровых и DDS/DAT-лентопротяжных устройств образуют дешевый сегмент рынка ленточных накопителей. Еще не так давно наибольшей популярностью пользовались 8-миллиметровые лентопротяжные устройства компании Exabyte, но в них наблюдалась тенденция к нарушению центровки каждые 6–12 месяцев, что вело к необходимости их регулировки в ремонтной мастерской. Часто лентопротяжный механизм растягивал ленты, и они становились нечитаемыми. Емкость этих лент, равная 2–7 Гбайт, делает их малопригодными для резервирования даже современных настольных систем, не говоря уже о серверах.

Накопители DDS/DAT (Digital Audio Tape — цифровая аудиолента) — это устройства, в которых запись данных осуществляется на 4-миллиметровую ленту методом спиральной развертки (наклонно-строчная запись). В действительности DAT-устройства относятся к стандарту DDS (Digital Data Storage — хранилище цифровых данных), но это непринципиальное различие. Исходный формат обеспечивал емкость около 2 Гбайт, но в следующих версиях стандарта DDS емкость существенно возросла. Последняя версия (DAT 72) позволяет хранить до 32 Гбайт данных при скорости их передачи равной 3,5 Гбайт/с. По заявлениям изготовителей ленты должны выдерживать до 100 циклов резервной записи, а их срок службы составляет до 10 лет.

Устройства DLT/S-DLT

Устройства Digital Linear Tape (цифровая лента с линейной записью) (DLT)/Super Digital Linear Tape (S-DLT) представляют основную разновидность носителей резервного копирования. Они надежны и позволяют хранить большие объемы данных. Их родоначальниками являются кассетные накопители TK-50 и TK-70 компании DEC. Впоследствии компания DEC продала технологию фирме Quantum, которая увеличила скорость записи и емкость носителей и снизила на них цены. В 2002 г. Quantum приобрела лицензию на технологию Super DLT, разработанную компанией Benchmark Storage Innovations. При использовании этой технологии записывающая головка наклоняется вперед и назад, что позволяет снизить перекрестные искажения между соседними дорожками.

В настоящее время компания Quantum предлагает две серии устройств: одну ориентированную на достижение максимальной производительности, а вторую — на максимальную емкость. Пользователь имеет возможность выбрать то, что ему наиболее подходит. Емкости ленты варьируются от 800 Гбайт в устройствах серии DLT-4 до 160 Гбайт в устройствах DLT-4 серии повышенной емкости при скоростях передачи данных соответственно равных 60 и 10 Мбайт/с. По заявлениям производителей, ленты прослужат от 20 до 30 лет. Но будут ли к тому времени существовать устройства, способные их прочесть? Много ли вам известно устройств чтения 9-дорожечных лент, которые до сих пор функционируют?

Недостатком технологии S-DLT является цена носителей, которая достигает $30–45. Для какой-нибудь инвестиционной компании с Уолл-стрит это, может быть, и нормальная цена, но для университета она несколько высоковата.

Устройства AIT и SAIT

Технология AIT (Advanced Intelligent Tape — усовершенствованная лента со встроенной логикой) — это линия усовершенствованных устройств записи на 8-миллиметровую ленту, выпускаемых компанией Sony. В 1996 г. компания отказалась от поддержки устройств Exabyte и представила собственный стандарт AIT-1, в котором также использовался метод наклонно-строчной записи (спиральная развертка), но емкость накопителей была в два раза выше. В настоящее время Sony предлагает версию AIT-1 повышенной емкости и свою последнюю разработку AIT-4, в которой емкость кассеты составляет 200 Гбайт при максимальной скорости передачи равной 24 Мбайт/с.

Предлагаемые компанией Sony устройства SAIT половинной высоты используют более широкий носитель и обладают большей емкостью, чем AIT. Ленты SAIT позволяют хранить до 500 Гбайт данных при скорости их передачи равной 30 Мбайт/с. Эти устройства наиболее популярны в качестве ленточных библиотек.

В AIT- и SAIT-устройствах используются ленты AME (Advanced Metal Evaporated — усовершенствованная технология напыления металла) с большим сроком эксплуатации. В ленточных картриджах имеется встроенное ЭСППЗУ (электрически стираемое программируемое ПЗУ), содержащее микрокод носителя, но для использования этого микрокода требуется программная поддержка. Цена устройств и лент сопоставима с ценой DLT-аналогов.

Устройства VXA/VXA-X

В настоящее время компания Exabyte предлагает устройства, в которых использованы технологии VXA и VXA-X. В лентопротяжных устройствах VXA использована технология, которая в компании Exabyte получила название пакетной технологии передачи данных. В устройствах VXA-X по-прежнему используются ленты AME производства компании Sony. Емкость устройств серии V возрастает с появлением носителей более высокой емкости. Заявленная емкость устройств серий VXA и X колеблется в диапазоне 33–160 Гбайт при скорости передачи равной 24 Мбайт/с.

Устройства LTO

Открытый стандарт ленты с линейной записью (Linear Tape-Open — LTO) был разработан компаниями IBM, HP и Quantum в качестве альтернативы запатентованному формату DLT. Последняя версия этого формата, LTO-3, позволяет хранить 400 Гбайт данных при скорости передачи равной 80 Мбайт/с. Предполагаемый срок эксплуатации лент LTO — 30 лет, но они чувствительны к влиянию магнитных полей. Цена картриджей емкостью 400 Гбайт составляет около $80.

Системы с автоматической загрузкой носителей

Стоимость современных жестких дисков столь мала, что многие организации могут позволить себе использовать диски огромной емкости. Поэтому для резервного копирования всех имеющихся данных иногда требуется несколько лент, даже если емкость каждой из них составляет 100 Гбайт. В таких случаях можно порекомендовать приобрести автозагрузчик, ленточный массив или библиотеку.

Автозагрузчик — это простое устройство для автоматической смены лент, используемое со стандартным накопителем. Он оснащен съемным магазином, в который помещаются ленты. Заполненная лента изымается из накопителя и заменяется пустой лентой, которую автозагрузчик берет из магазина. В магазинах автозагрузчиков обычно помещается до десяти лент.

Ленточный массив — это устройство, которое автоматически меняет съемные носители в нескольких накопителях. Выпускаются массивы для различных типов носителей. Часто они поставляются вместе со специальными программами создания резервных копий, которые манипулируют устройством смены носителей. Такие программные продукты выпускаются, в частности, компаниями Storage Technology (теперь Sun Microsystems) и Sony.

Ленточные библиотеки — это устройства для хранения огромных (терабайтовых) объемов данных. Они представляют собой аппаратные комплексы размером со шкаф, в которых имеется манипулятор типа "рука", обслуживающий многочисленные полки с ленточными носителями или компакт-дисками. Несложно догадаться, что эти устройства очень дороги как при покупке, так и в обслуживании, ведь им требуется специальное электропитание, помещение и особый режим вентиляции. Как правило, вместе с библиотекой заказываются услуги оператора, который отвечает за установку и наладку комплекса. С библиотекой поставляется также программный пакет, управляющий ее работой. Ведущим производителем ленточных библиотек является компания Storage Technology (Sun Microsystems).

Жесткие диски

Стремительное снижение стоимости жестких дисков делает их вполне достойными кандидатами на роль устройств резервного копирования. И хотя мы не рекомендуем копировать один диск на другой на том же самом компьютере, но можно создавать резервные копии по сети; их стоимость при этом будет очень невелика.

Конечно, емкость жесткого диска ограничена, и со временем он будет занят полностью. Однако резервные копии на жестких дисках — прекрасное средство защиты от случайного удаления файлов. Когда есть вчерашний образ диска, доступный по сети через NFS или CIFS, пользователи получают возможность исправлять свои ошибки без вмешательства администратора. Еще один популярный метод резервирования систем индивидуальных настольных компьютеров — использование современных FireWire и USB-дисков высокой емкости. В частности, 1-кнопочное устройство немедленного резервного копирования ZDisc компании Cintre, представляющее собой внешний жесткий диск форм-фактора 3,5 дюйма с интерфейсом USB 2.0, стоимостью около $100 — весьма привлекательное решение.

Типы носителей

Как мы видим, существует множество возможностей. В табл. 9.1 приведены характеристики рассмотренных выше носителей и соответствующих им устройств.

Кертис Престон (W. Curtis Preston) публикует в Internet список устройств резервного копирования с указанием их характеристик и производителей. Этот список доступен по адресу `www.backupcentral.com/hardware-drives.html`.

Таблица 9.1. Сравнительные характеристики носителей, применяемых для резервного копирования

Носитель	Емкость[а]	Скорость[а]	Цена накопителя	Цена носителя	Цена в расчете на 1 Гбайт	Многократное использование?	Произвольный доступ?[б]
CD-R	650 Мбайт	4 Мбайт/с	$15	$0,20	$0,32	Нет	Да
CD-RW	650 Мбайт	4 Мбайт/с	$20	$0,30	$0,48	Да	Да
DVD±R	4,7 Гбайт	10 Мбайт/с	$50	$40	$0,09	Нет	Да
DVD+R DL[в]	8,5 Гбайт	10 Мбайт/с	$50	$2	$0,24	Нет	Да
DVD±RW	4,7 Гбайт	10 Мбайт/с	$50	$0,80	$0,17	Да	Да
DDS-4 (4 мм)	20 Гбайт	10 Мбайт/с	$300	$5	$0,25	Да	Нет
DLT/S-DLT	160 Гбайт	10 Мбайт/с	$1000	$20	$13	Да	Нет
AIT-4 (8 мм)	200 Гбайт	6 Мбайт/с	$2500	$40	$0,20	Да	Нет
SAIT-1	500 Гбайт	30 Мбайт/с	$6500	$126	$0,25	Да	Нет
VXA-172	86 Гбайт	12 Мбайт/с	$600	$30	$0,35	Да	Нет
VXA-320	160 Гбайт	12 Мбайт/с	$800	$60	$0,35	Да	Нет
LTO - 3	400 Гбайт	30 Мбайт/с	$3000	$65	$0,16	Да	Нет

[а] Емкость и скорость указаны без учета сжатия данных.

[б] Допускает произвольный доступ к любому фрагменту данных?

[в] Двухслойный.

Что покупать

Таблица 9.1 достаточно точно отражает состояние рынка оборудования для выполнения резервного копирования. Все устройства резервного копирования работают достаточно хорошо, и среди систем одной ценовой категории обычно трудно отдать чему-либо предпочтение. Покупайте устройство, которое отвечает требованиям вашей организации и имеет приемлемую цену.

Устройства DDS, AIT и LTO лучше всего подходят для небольших рабочих групп и отдельных компьютеров, в которых установлены диски высокой емкости. Начальная цена этих систем относительно умеренна, а их носители широко распространены, к тому же за каждым из стандартов стоит целый ряд производителей. Все системы обладают достаточным быстродействием, что позволяет архивировать большие объемы данных за разумное время.

Устройства DLT, AIT и LTO-3 имеют сопоставимые характеристики. Трудно сделать выбор в пользу одной из этих технологий, тем более что ситуация может измениться с появлением новых спецификаций и моделей устройств. Все эти форматы тщательно продуманы и могут быть успешно интегрированы в любую систему — будь то среда учебного заведения или среда крупной коммерческой организации.

В следующих разделах базовый термин “лента” используется в качестве универсального обозначения носителей, выбранных для резервного копирования. Примеры команд, выполняющих резервное копирование, даются в контексте ленточных накопителей.

9.3. Команда dump: настройка режима инкрементного архивирования

Самые распространенные программные средства создания резервных копий и восстановления данных — команды `dump` и `restore`. Они существуют уже длительное время и хорошо изучены. В большинстве организаций команды `dump` и `restore` используются автоматизированными системами резервного копирования.

Присутствие команд `dump` и `restore` в системе зависит от того, какие опции были заданы при установке Linux, поэтому не исключено, что их придется инсталлировать явно. Для каждой из систем существует специальный программный пакет, упрощающий процедуру установки. Текущие версии ядра систем Red Hat и Fedora во время установки предлагают установить административный пакет, который включает в себя команду `dump`.

В Linux ничто не компонуется статически, поэтому необходимо также наличие в каталоге `/lib` соответствующих совместно используемых библиотек.

Архивирование файловых систем

Команда `dump` формирует перечень файлов, которые модифицировались с момента предыдущего архивирования, а затем упаковывает эти файлы в один большой архив, подлежащий записи на внешнее устройство. Команда `dump` обладает рядом преимуществ по сравнению с другими утилитами, описанными в этой главе:

- резервные копии могут быть записаны на несколько лент;
- можно выполнять резервное копирование и восстановление файлов любого типа (даже файлов устройств);
- права доступа, информация о владельцах и даты модификации файлов сохраняются;
- обеспечивается правильная обработка файлов с “дырами”[4];
- резервное копирование может выполняться в инкрементном режиме (на ленту записываются только модифицированные версии файлов).

GNU-версия команды `tar`, используемая в Linux, также реализует все эти возможности, но средства инкрементного архивирования у команды `dump` лучше.

Хотя на самом деле причина, по которой в среде Linux предпочтение отдается команде `dump`, заключается совсем в другом. К сожалению, так сложилось, что версия команды `tar`, входящая в состав большинства основных UNIX-систем, не обладает вышеперечисленными особенностями GNU-версии. Если с резервными копиями нужно работать не только в Linux, но и в других UNIX-системах, то у команды `dump` нет конкурентов. Она одинаково ведет себя на разных платформах, поэтому нет смысла осваивать две команды, когда достаточно одной. Команда `dump` предоставляет меньше возможностей, но `tar` разборчива!

[4] “Дыры” — это блоки, которые никогда не содержали данных. Если открыть файл и записать в него один байт, а затем переместить указатель текущей позиции на 1 Мбайт вперед и записать еще один байт, то полученный “разреженный” файл будет занимать всего два дисковых блока, хотя его логический размер гораздо больше. Много “дыр” обычно содержат файлы, созданные в СУБД Berkeley DB или `ndbm`.

Команда dump понимает структуру исходных файловых систем и непосредственно читает таблицы индексных дескрипторов, чтобы определить, какие файлы подлежат архивированию. Такое знание файловой системы делает команду dump весьма эффективной, но вместе с тем налагает на нее определенные ограничения[5].

🕮 Подробная информация об NFS приведена в главе 16.

Одно из ограничений заключается в том, что каждая файловая система должна архивироваться в индивидуальном порядке. Если диск разбит на разделы, придется копировать каждый из них отдельно. Другое ограничение: разрешается копировать только файловые системы локального компьютера, а файловую систему NFS, смонтированную по сети, архивировать нельзя. Тем не менее, можно создать резервную копию локальной файловой системы на удаленном ленточном накопителе; для этого используется команда rdump.

Важнейшая особенность команды dump — поддержка инкрементного резервного копирования. Можно, конечно, каждый день создавать копии всей системы, но это не совсем практично. В инкрементном режиме архивируются только те файлы, которые изменились с момента последнего копирования.

При создании архива ему присваивается номер уровня (целое число от 0 до 9). В архив уровня *N* копируются все файлы, которые изменились с момента создания последнего архива уровня ниже *N*. В архив нулевого уровня включается вся файловая система. В режиме инкрементного архивирования файловую систему можно вернуть в то состояние, в котором она находилась в момент последнего резервного копирования, хотя для этого может потребоваться восстановление файлов из нескольких лент[6].

Еще одна приятная особенность команды dump связана с тем, что она не обращает внимания на длину имен файлов. Иерархии каталогов могут быть произвольно глубокими, и длинные составные имена обрабатываются корректно.

Первым аргументом команды dump должен быть уровень архива. Для того чтобы определить, насколько давно создавался последний архив, команда dump проверяет файл /etc/dumpdates. Флаг -u заставляет команду по завершении копирования автоматически обновить файл /etc/dumpdates. При этом регистрируются дата, уровень архива и имя файловой системы. Если флаг -u ни разу не использовался, всем архивам будет присваиваться уровень 0, поскольку факт предыдущего резервного копирования файловой системы нигде не отражался. Если имя файловой системы изменилось, можно отредактировать файл /etc/dumpdates вручную.

🕮 О номерах устройств рассказывается в разделе 28.2.

Свою выходную информацию команда dump посылает на устройство, заданное по умолчанию. Как правило, это основной ленточный накопитель. Для указания другого устройства необходимо использовать флаг -f. Когда на одну ленту помещается несколько архивов, задайте имя ленточного устройства, не поддерживающего режим обратной перемотки (т.е. укажите файл устройства, который не вызывает перемотку ленты по окончании записи — для большинства ленточных устройств такой файл создается наряду со стандарт-

[5] Команда dump требует доступа к неструктурированным разделам диска. Каждый, кому разрешено выполнять резервное копирование, может, приложив определенные усилия, прочесть все файлы в системе.

[6] Команда dump не отслеживает, какие файлы были удалены с момента создания архива. При восстановлении информации из инкрементных резервных копий удаленные файлы снова появятся в системе.

ным)[7]. В Linux режиму обратной перемотки первого ленточного накопителя SCSI обычно соответствует файл /dev/st0, а режиму без перемотки — файл /dev/nst0.

Если случайно выбран файл устройства с обратной перемоткой, все закончится сохранением только последней заархивированной файловой системы. Поскольку команда dump не проверяет, перемотана лента в начало или нет, эта ошибка не повлечет за собой других ошибок и станет очевидной лишь при попытке восстановить файлы.

Если архив помещается в удаленную систему с помощью команды rdump, удаленное устройство нужно задать в формате *имя_компьютера:устройство*, например:

```
# rdump -0u -f anchor:/dev/nst0 /spare
```

Права доступа к удаленному ленточному устройству определяются посредством канала SSH. Подробнее о SSH рассказывается в разделе 20.11.

Раньше команде dump нужно было указывать точную длину ленты, чтобы запись прекращалась при достижении конца ленты. Современные накопители самостоятельно обнаруживают конец ленты и сообщают об этом команде dump, которая перематывает ленту в начало, выталкивает носитель из устройства и запрашивает новую ленту. Поскольку применение различных аппаратных алгоритмов сжатия не позволяет определить длину ленты на основании емкости носителя, лучше ориентироваться на сигнал конца ленты (EOT — End Of Tape), если, конечно, его поддерживает устройство. В противном случае задайте длину ленты в килобайтах с помощью опции -B.

```
# dump -5u -B 2000000 -f /dev/nst0 /work
DUMP: Date of this level 5 dump: Wed May   8 16:59:45 2006
DUMP: Date of last level 0 dump: the epoch
DUMP: Dumping /dev/hda2 (/work) to /dev/nst0
DUMP: mapping (Pass I) [regular files]
DUMP: mapping (Pass II) [directories]
DUMP: estimated 18750003 tape blocks on .23 tape(s)
....
```

За флагами -5u следуют флаги -B (емкость ленты в килобайтах: 20 Гбайт) и -f (файл устройства: /dev/nst0). В конце указывается обязательный аргумент: имя файловой системы (/work).

Схемы создания архивов

Поскольку уровни архивов имеют смысл только в связи с другими уровнями, то существуют самые разные схемы создания резервных копий.

Выбор схемы зависит от следующих факторов:

- от активности файловых систем;
- от емкости устройства резервного копирования;
- от необходимой степени избыточности;
- от числа лент, которые нужно приобрести.

Раньше, когда для резервного копирования файловых систем требовалось много лент, разрабатывались сложные графики архивирования, позволявшие свести к минимуму число лент, необходимых ежедневно. Сегодня емкость лент существенно возросла, поэтому разница между уровнями архивов уже не так важна.

[7] Для всех файлов ленточного устройства задан один и тот же старший номер устройства. Младший номер устройства сообщает драйверу о наличии особых свойств (режим обратной перемотки, режим перестановки байтов и т.д.).

Поскольку большинство файлов никогда не меняется, даже в простейшей схеме инкрементного архивирования многие файлы не включаются в ежедневный архив. При введении дополнительных уровней небольшое число регулярно изменяющихся файлов дробится на более мелкие сегменты.

Сложная схема резервного копирования обеспечивает следующие три преимущества:

- данные будут архивироваться чаще, что позволит снизить возможные потери;
- ежедневно будет использоваться меньшее число лент (или же все поместится на одну ленту);
- в целях защиты от ошибок будет создаваться несколько копий каждого файла.

В целом, выбор схемы резервного копирования определяется потребностями в названных областях. Мы опишем несколько возможных схем и приведем аргументы в пользу каждой из них. Иногда можно использовать эти схемы без изменений, а иногда приходится разрабатывать совершенно другие схемы.

Простая схема

Если общий размер дискового пространства меньше емкости ленты, можно предложить простой график архивирования. Архивы нулевого уровня каждой файловой системы должны создаваться ежедневно. Используйте многократно один набор лент, но через каждые *N* дней (где *N* определяется потребностями организации) откладывайте ленты навсегда. Стоимость такой схемы составит

```
(365/N) * (цена ленты)
```

в год. При создании очередного архива нельзя использовать предыдущую ленту. Лучше осуществлять ротацию лент, чтобы в случае уничтожения одного из архивов можно было восстановить архив предыдущего дня.

Подобная схема обеспечивает высокую избыточность и значительно упрощает восстановление данных. Она хорошо подходит для организации, имеющей много денег, но очень занятого (или неопытного) администратора. С точки зрения надежности и удобства это идеальная схема. От нее не следует отступать без особой на то причины (например, в целях экономии лент или снижения трудоемкости).

Умеренная схема

В большинстве организаций придерживаются более разумного подхода: выделяется по одной ленте на каждый день недели, каждую неделю месяца (понадобится пять таких лент) и каждый месяц года. Ежедневно создается дневной архив девятого уровня, еженедельно — недельный архив пятого уровня, ежемесячно — месячный архив третьего уровня. Архив нулевого уровня нужно создавать всякий раз, когда инкрементные копии становятся слишком большими для одной ленты. Чаще всего это происходит с лентами месячного архива. В любом случае архивирование нулевого уровня необходимо проводить хотя бы раз в год.

Выбор уровней 3, 5 и 9 сделан произвольно. С таким же успехом можно использовать уровни 1, 2 и 3, но интервалы между уровнями дают определенную свободу для маневра на случай, если впоследствии потребуется добавить еще один уровень.

Такой график требует наличия двадцати четырех основных лент, а также какого-то количества лент для архивов нулевого уровня. Общее число необходимых лент невелико, но и избыточность невысока.

9.4. Команда restore: восстановление файлов из резервных копий

Команда, восстанавливающая данные с резервных лент, называется `restore`. Вначале мы рассмотрим восстановление отдельных файлов (или небольших групп файлов), а затем — файловых систем целиком.

Восстановление отдельных файлов

Первое, что нужно сделать, узнав о пропаже файла, — выяснить, на каких лентах есть его версии. Чаще всего пользователи просят найти самую последнюю версию файла, но так бывает не всегда. Например, пользователь, потерявший файл из-за случайного копирования поверх него другого файла, будет искать ту версию, которая существовала до этой неприятности. Желательно, чтобы пользователи сообщали не только о том, какие файлы пропали, но и когда они пропали и когда модифицировались последний раз. Имеет смысл формализовать сообщения пользователей с помощью опросной формы.

Если не ведутся оперативные таблицы архивов, придется монтировать ленты одну за другой и пытаться восстанавливать пропавшие файлы, пока не будет найдена нужная лента. Если пользователь помнит примерную дату последнего изменения файла, можно с достаточной степенью вероятности предположить, на какой ленте он находится.

Определив ленты, с которых будет производиться восстановление, создайте временный каталог, к примеру, `/var/restore`, где будет сформирована большая иерархия подкаталогов, и перейдите в него с помощью команды `cd`. Команда `restore` обычно создает каталоги, ведущие к файлу, прежде чем восстанавливать его. Не используйте для этих целей каталог `/tmp`: в случае системного сбоя и последующей перезагрузки содержимое каталога может быть стерто.

У команды `restore` много опций. Самые важные из них — `-i`, которая позволяет восстанавливать отдельные файлы и каталоги в интерактивном режиме, и `-r`, задающая полное восстановление всей файловой системы. Опция `-x` запрашивает автоматическое восстановление указанных файлов — будьте осторожны, чтобы не затереть существующие файлы.

Команда `restore -i` читает с ленты таблицу содержимого, а затем позволяет перемещаться по архиву, как по обычному дереву каталогов, с помощью команд `ls`, `cd` и `pwd`. Найдя файл, который нужно восстановить, выполните команду `add` *`имя_файла`*. Когда все необходимые файлы выбраны, скопируйте их с ленты командой `extract`.

📖 Команда `mt` описана в разделе 9.7.

Если на одной ленте находится много архивов, то перед выполнением команды `restore` необходимо перемотать ленту на соответствующий архив с помощью команды `mt`. Не забудьте выбрать файл устройства без режима обратной перемотки!

Например, для восстановления файла `/users/janet/iamlost` с удаленного ленточного накопителя необходимо задать показанную ниже последовательность команд. Предполагается, что нужная лента найдена и смонтирована в точке `tapehost:/dev/nst0`, а файловая система, содержащая начальный каталог пользователя janet, — четвертая на ленте.

```
# mkdir /var/restore
# cd /var/restore
```

```
# ssh tapehost mt -f /dev/nst0 fsf 3
# rrestore -i -f tapehost:/dev/nst0
restore> ls
.:
janet/ garth/ lost+found/ lynda/
restore> cd janet
restore> ls
afile bfile cfile iamlost
restore> add iamlost
restore> ls[8]
afile bfile cfile iamlost*
restore> extract
You have not read any volumes yet.
Unless you know which volume your files are on you should
start with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] n
```

Ленточные тома нумеруются с единицы, а не с нуля, поэтому для архива, который умещается на одной ленте, нужно указать значение 1. Спрашивая, следует ли задать имя владельца и режим доступа для каталога ".", команда `restore` "интересуется", должен ли текущий каталог соответствовать корневому каталогу ленты. Обычно в этом нет необходимости, если только не восстанавливается вся файловая система целиком.

После того как команда `restore` завершит работу, нужно передать извлеченный файл пользователю janet:

```
# cd /var/restore
# ls janet iamlost
# ls ~janet
afile bfile cfile
# cp -p janet/iamlost ~janet/iamlost.restored
# chown janet ~janet/iamlost.restored
# rm -rf /var/restore
# mail janet
Your file iamlost has been restored as requested and has been placed in
/users/janet/iamlost.restored.

Your name, Humble System Administrator
```

Некоторые администраторы восстанавливают файлы в специальный каталог и дают пользователям возможность самостоятельно копировать их в свои каталоги. В этом случае необходимо обеспечить защиту файлов, назначив им соответствующих владельцев и нужные права доступа. Не забывайте очищать этот каталог, чтобы файлы не "засоряли" систему.

Если резервная копия создавалась с помощью команды `rdump` и команда `restore` не может восстановить файлы из архива, попробуйте воспользоваться командой `rrestore`. Чтобы свести к минимуму вероятность возникновения проблем, старайтесь читать ленту на том же компьютере, где она записывалась.

Команда `restore -i` — самый простой способ восстановления нескольких файлов или каталогов из архива. Но она не будет работать, если ленту нельзя перематывать по одной записи назад (такая проблема существует у некоторых накопителей на 8-миллиметровых лентах). В таком случае попробуйте использовать команду `restore -x`. Она

[8] Звездочка рядом с именем `iamlost` означает, что этот файл помечен для восстановления.

требует указания полного имени восстанавливаемого файла (относительно корневого каталога архива) в командной строке. Следующая группа команд повторяет показанный выше пример:

```
# mkdir /var/restore
# cd /var/restore
# ssh tapehost mt -f /dev/nst0 fsf 3
# rrestore -x -f tapehost:/dev/nst0 ./janet/iamlost
```

Восстановление файловых систем

Есть счастливчики, которым никогда не приходилось после серьезного сбоя восстанавливать всю файловую систему. Другим везет меньше. Прежде чем пытаться восстанавливать файловую систему, убедитесь, что проблема, которая привела к ее разрушению, устранена. Глупо часами перематывать ленты лишь для того, чтобы тут же потерять файловую систему еще раз.

Перед началом процедуры восстановления нужно создать и смонтировать целевую файловую систему. О том, как это делается, подробно рассказывалось в главе 7. Далее перейдите в каталог монтирования новой файловой системы, вставьте в накопитель первую ленту самого последнего архива нулевого уровня и введите `restore -r`.

Команда `restore` будет сама подсказывать, когда нужно вставить следующую ленту. Восстановив архив нулевого уровня, повторяйте процедуру для инкрементных архивов в том порядке, в каком они создавались. Поскольку всегда существует определенная избыточность, то, как правило, нет необходимости восстанавливать все инкрементные архивы. Вот примерная последовательность действий.

Шаг 1: восстановите самый последний архив нулевого уровня.

Шаг 2: восстановите тот из оставшихся архивов, у которого наименьший уровень; если на данном уровне было создано несколько архивов, восстановите новейший из них.

Шаг 3: если это оказался самый последний архив, процедуру можно считать завершенной.

Шаг 4: в противном случае вернитесь к шагу 2.

Приведем несколько примеров архивных последовательностей. Восстанавливать нужно только те архивы, номера которых выделены полужирным шрифтом:

```
0 0 0 0 0 0
0 5 5 5 5
0 3 2 5 4 5
0 9 9 5 9 9 3 9 9 5 9 9
0 3 5 9 3 5 9
```

О командах `mke2fs` и `mount` рассказывалось в главе 7.

Давайте проанализируем все используемые команды. Например, если последним был создан ежемесячный архив уровня 3, а перед ним создавался ежегодный архив нулевого уровня (см. раздел “Умеренная схема” выше), то для восстановления файловой системы `/home`, находящейся на физическом устройстве `/dev/sda1`, понадобятся следующие команды (имена устройств зависят от типа оборудования):

```
# /etc/mke2fs -j /dev/sda1 QUANTUM_PD1050S
# /etc/mount /dev/sda1 /home
```

```
# cd /home
/* Монтируем первую ленту архива уровня 0 для каталога /home. */
# restore -r
/* Монтируем ленты, запрашиваемые командой restore. */
/* Монтируем первую ленту ежемесячного архива уровня 3. */
# restore -r
```

Если на одной резервной ленте находится несколько файловых систем, то перед обращением к команде `restore` необходимо с помощью команды `mt` перемотать ленту на нужную файловую систему. Команда `mt` описана в разделе 10.7.

Приведенная последовательность команд позволит восстановить файловую систему в том состоянии, в котором она находилась, когда был создан архив третьего уровня, с одной особенностью: заодно будут "воскрешены" все удаленные с тех пор файлы. Эта проблема особенно неприятна, когда восстанавливается активная файловая система или диск практически заполнен. Не исключено, что во втором случае команда `restore` завершится неудачно.

9.5. Архивирование и восстановление при модификации операционной системы

В процессе обновления операционной системы необходимо путем создания архива нулевого уровня получить резервные копии всех файловых систем, причем иногда их приходится сразу восстанавливать. Последнее требуется лишь в том случае, если в новой ОС используется другой формат файловой системы или если изменяется структура разделов диска. В то же время резервное копирование *нужно обязательно* предусматривать как меру предосторожности на случай проблем, которые могут возникнуть в процессе модификации. Кроме того, наличие полного комплекта резервных копий даст возможность переинсталлировать старую операционную систему, если новая версия не подойдет. К счастью, благодаря применению систем последовательного обновления, используемых в большинстве современных дистрибутивов, эти ленты вряд ли потребуются.

Не забудьте скопировать на ленту и восстановить все системно-зависимые файлы в каталогах / и `/usr`, такие как `/etc/passwd`, `/etc/shadow`, `/usr/local` и т.д. Эта довольно трудная задача, ибо организация каталогов, принятая в Linux, приводит к перемешиванию файлов, созданных на компьютере, с файлами, входящими в системный дистрибутив.

Сразу после завершения модификации нужно еще раз создать полный набор архивов нулевого уровня. Процедуры модернизации, принятые в большинстве операционных систем, предполагают установку даты модификации системных файлов по времени их создания производителем системы, а не по текущему времени. Это означает, что в случае сбоя инкрементные архивы, которые были созданы относительно архива нулевого уровня, записанного до модернизации, окажутся недостаточными для восстановления системы в модернизированном состоянии.

9.6. Другие архиваторы

Команда `dump` является наиболее эффективным средством архивирования файловых систем, хотя это не единственная команда, которую можно использовать для записи файлов на ленты. Команды `tar`, `cpio` и `dd` тоже умеют перемещать файлы с одного носителя на другой.

Команда tar: упаковка файлов

Команда tar объединяет несколько файлов или каталогов в один файл, часто записываемый прямо на ленту. Это удобный инструмент создания резервных копий файлов, которые предполагается восстанавливать в ближайшем будущем. Например, если пользователь уезжает на полгода, а в системе мало места на диске, администратор может воспользоваться командой tar и перенести файлы этого пользователя на ленту, после чего удалить их с диска.

Команда tar хорошо подходит для перемещения дерева каталогов, особенно если файлы копируются от имени пользователя root. Команда tar сохраняет информацию о принадлежности объектов и времени их создания, но только если это указано в ее параметрах. Например, команда

```
tar -cf - исходный_каталог | (cd целевой_каталог ; tar --atime-preserve -xpf -)
```

создает копию дерева исходного каталога в целевом каталоге. В аргументе *целевой_каталог* следует избегать использования символов "..", поскольку символьные ссылки и команды автоматического монтирования могут трактовать эту последовательность несколько иначе. Нам доводилось сталкиваться с подобной ситуацией.

По умолчанию команда tar не выполняет разрешение символьных ссылок, но может делать это по указанию. Можно также заставить ее включать в архив только те файлы, которые были модифицированы после заданной даты, что весьма удобно для организации инкрементного копирования. Описание этих и других полезных свойств можно найти на man-странице команды.

Для некоторых версий команды tar (не в Linux) характерна одна проблема: длина имени файла по умолчанию ограничена 100 символами. Это не позволяет использовать команду для архивирования глубоких иерархий каталогов. Если в Linux создаются tar-архивы, которыми предполагается делиться с пользователями других систем, помните о том, что обладатели стандартной команды tar могут не суметь прочитать их.

Опция -b команды tar позволяет задать коэффициент объединения блоков, который должен учитываться при записи информации на ленту. Этот коэффициент указывается в виде числа 512-байтовых фрагментов и определяет, какой объем данных команда помещает во внутренний буфер перед выполнением записи. Некоторые DAT-устройства начинают работать некорректно, если используется нестандартный коэффициент объединения, тогда как другие накопители его игнорируют.

В ряде систем при определенных значениях коэффициента объединения блоков производительность работы с лентой повышается. Оптимальный коэффициент зависит от конкретного компьютера и ленточного накопителя, хотя во многих случаях разница в быстродействии незаметна. Если есть сомнения, задайте коэффициент 20.

Linux-версия команды tar заполняет "дыры" в файлах, если только при создании исходного архива не была указана опция -S. Эта версия плохо справляется с наличием ошибок на лентах.

Команда cpio: древний архиватор

Команда cpio по своим функциональным возможностям близка к команде tar. Эта команда использовалась в первых версиях операционных систем и в настоящее время применяется редко. Но ее можно применять для переноса дерева каталогов. Команда

```
find исходный_каталог -depth -print | cpio -pdm целевой_каталог
```

создает копию дерева исходного каталога в целевом каталоге. GNU-версия команды cpio, используемая в Linux, позволяет записывать архивы на несколько лент, хотя другие версии этой возможностью не обладают. Специальные файлы может копировать только пользователь root. Опции этой команды сильно отличаются в разных системах, поэтому, даже если раньше вам доводилось иметь дело с командой cpio, прочтите предварительно ее man-страницу.

Команда dd: манипулирование битами

Команда dd предназначена для копирования и преобразования файлов. При отсутствии иных указаний команда просто копирует информацию из входного файла в выходной. Если пользователь принес ленту, которая была записана не в Linux, использование команды dd может оказаться единственным способом ее прочитать.

Одним из первоначальных применений команды dd было создание копии всей файловой системы. Сегодня есть более эффективный вариант: создать с помощью команды mke2fs целевую файловую систему, а затем выполнить команду dump, соединив ее каналом с командой restore. При неправильном использовании команда dd может искажать информацию о структуре разделов. Она копирует файловые системы только между разделами одинакового размера.

Команду dd можно также применять для создания копий магнитных лент. При наличии двух ленточных накопителей (к примеру, /dev/st0 и /dev/st1) используйте команду

```
$ dd if=/dev/st0 of=/dev/st1 cbs=16b
```

Если есть только один накопитель (/dev/st0), введите такую последовательность:

```
$ dd if=/dev/st0 of=tfile cbs=16b
/* Меняем ленты */ */
 $ dd if=tfile of=/dev/st0 cbs=16b $ rm tfile
```

Конечно, когда имеется всего один накопитель, на диске должно быть достаточно места для сохранения образа ленты.

9.7. Запись нескольких архивов на одну ленту

Магнитная лента фактически содержит одну длинную строку данных. Но очень часто возникает необходимость поместить на ленту несколько архивов, поэтому ленточные накопители и их Linux-драйверы на логическом уровне реализуют более сложную структуру хранения. Когда команда dump (или ее аналог) записывает поток байтов на ленту, а затем закрывает файл устройства, на ленту автоматически помещается маркер конца файла (EOF — end of file). Этот маркер отделяет один поток от другого. При извлечении потока данных чтение автоматически прекращается в случае обнаружения маркера EOF.

Чтобы перемотать ленту для записи определенного потока, используется команда mt. Она очень удобна при размещении на одной ленте нескольких архивов. Кроме того, ее сообщения об ошибках — одни из самых интересных среди утилит Linux. Базовый формат команды таков:

```
mt [-f имя_ленты] инструкция [счетчик]
```

Аргумент *инструкция* может иметь множество значений. От платформы к платформе они меняются, поэтому мы рассмотрим лишь те варианты, которые важны для создания резервных копий и их восстановления.

`rew`	Перемотать ленту к началу.
`offl`	Перевести ленту в автономный режим. В большинстве накопителей это приводит к перемотке ленты в начало и выталкиванию ее из приемника. Сценарии архивирования применяют эту инструкцию для извлечения ленты по окончании операции, что должно свидетельствовать об успешном ее завершении
`status`	Вывести информацию о текущем состоянии ленточного накопителя (вставлена ли лента и т.д.).
`fsf [`*`счетчик`*`]`	Перемотать ленту вперед. Если аргумент *счетчик* отсутствует, лента перематывается на один файл. Если указан числовой аргумент, лента перематывается на соответствующее количество файлов. Эту инструкцию можно использовать для перехода к нужной файловой системе на ленте, содержащей несколько архивов
`bsf [`*`счетчик`*`]`	Перемотать ленту назад на указанное число файлов. Точная интерпретация инструкции зависит от типа накопителя и его драйвера. Иногда текущий файл учитывается, иногда — нет. В некоторых случаях эта инструкция не делает ничего (и не сообщает об этом). Если лента перемотана слишком далеко, лучше всего ввести инструкцию `mt rew` и начать сначала.

Для ознакомления с перечнем всех поддерживаемых команд обратитесь к man-странице команды `mt`.

Те, кому повезло иметь в своем распоряжении роботизированную ленточную библиотеку, могут воспользоваться пакетом `mtx`, который представляет собой улучшенную версию команды `mt` и позволяет манипулировать устройством смены лент. Мы применяли его для автоматизированного управления лентами в библиотеке DAT-картриджей HP 6x24.

9.8. Bacula

Bacula — это программа резервного копирования систем клиент/сервер на уровне организации, которая управляет резервным копированием, восстановлением и проверкой целостности файлов по сети. Она работает в большинстве UNIX- и Linux-систем, в том числе в описанных в этой книге дистрибутивах. Она может также выполнять резервное копирование других операционных систем, включая Microsoft Windows.

В предыдущем издании этой книги в качестве некоммерческого средства резервного копирования мы рекомендовали использовать программу Amanda. Информацию о ней можно найти в первом издании этой книги или по адресу `www.amanda.org`. Ниже приведены аргументы в пользу выбора программы Bacula.

- Она имеет модульную структуру.
- Она позволяет выполнять резервное копирование файловых систем UNIX, Linux и Windows.

- В качестве вспомогательной системы управления базой данных она поддерживает MySQL, PostgreSQL и SQLite.
- Эта программа поддерживает простую в использовании, управляемую с помощью меню консоль командной строки.
- Ее исходный код соответствует лицензии программного обеспечения с открытым исходным кодом.
- Ее резервные копии могут быть записаны на несколько ленточных томов.
- Серверы этой программы могут работать на нескольких платформах.
- Для каждого копируемого файла программа может создавать сигнатуры SHA1 или MD5.
- Программа проверяет результаты резервного копирования.
- Она поддерживает ленточные библиотеки и устройства автоматической замены лент.
- До и после выполнения задач резервного копирования она может выполнять сценарии и команды.
- Программа осуществляет централизованное управление резервным копированием для всей сети.

Модель, используемая программой Bacula

Чтобы использовать программу Bacula, необходимо понимать работу ее основных компонентов. Общая архитектура системы Bacula показана на рис. 9.1.

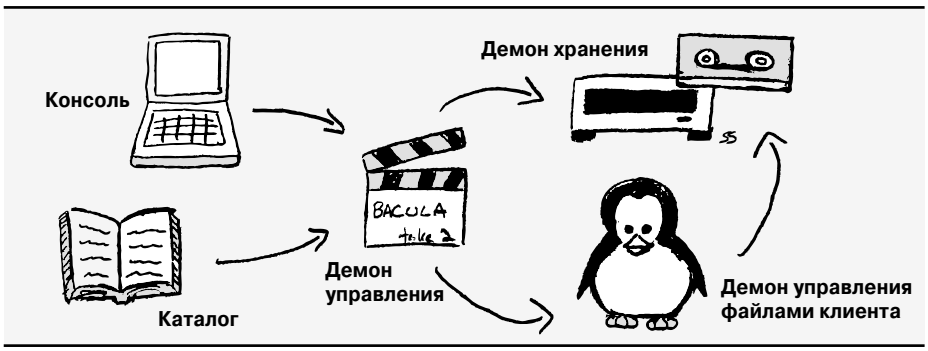


Рис. 9.1. Компоненты системы Bacula и их взаимосвязь

Демон управления Bacula — это демон, который координирует операции резервного копирования, восстановления и проверки. Передача заданий резервного копирования или восстановления демону управления осуществляется посредством консоли Bacula. Демон управления может также запрашивать демон хранения или демоны файловой службы, размещенные на компьютерах клиентов.

Обмен данными с демоном управления выполняется посредством консоли Bacula, которая может функционировать в качестве графического интерфейса пользователя GNOME либо в качестве средства командной строки. Консоль может быть запущена на любом компьютере, а не только на том, где действует демон управления.

Демон хранения — компонент системы Bacula, который выполняет чтение и запись лент и других носителей резервного копирования. Эта служба должна быть запущена на компьютере, к которому подключено ленточное устройство или устройство хранения, используемое для хранения резервных копий. Но ее установка на том же сервере, на котором установлен демон управления, не обязательна (хотя и допускается).

Демон управления файлами действует в каждой системе, файлы которой требуется копировать. Во время выполнения заданий резервного копирования реализации этого демона каждой из поддерживаемых операционных систем отправляют демону хранения данные и атрибуты соответствующих файлов.

Последний компонент системы Bacula — каталог, представляющий собой реляционную базу данных, в которой Bacula хранит информацию о каждом копируемом файле и томе. Он позволяет программе Bacula быстро и эффективно выполнять восстановление, поскольку ретроспективная информация о всех операциях резервного копирования доступна в оперативном режиме; программе известно, какие тома с резервными копиями требуются для восстановления конкретного набора файлов, еще до того, как она выполнит чтение хотя бы одной ленты. В настоящее время Bacula поддерживает три базы данных: MySQL, PostgreSQL и SQLite. Базе данных каталога не обязательно размещаться на том же сервере, на котором установлен демон управления.

Дополнительный, необязательный компонент — Bacula Rescue CD-ROM (аварийный компакт-диск Bacula). Он представляет собой отдельно загружаемый пакет, который создает индивидуальные загрузочные компакт-диски аварийного восстановления систем Linux. Эти компакт-диски содержат статически скомпонованную копию демона файлов системы и специализированные сценарий интерпретатора с информацией о конфигурации дисков системы, ядра и сетевых интерфейсов. При возникновении фатального сбоя системы Linux ее компакт-диски аварийного восстановления позволят выполнить загрузку системы, повторное разбиение диска на логические разделы и подключение демона управления Bacula для полного восстановления системы по сети.

Настройка программы Bacula

По причине сложности, поддержки расширенного набора функций и модульной структуры программы Bacula существует множество способов настройки схемы резервного копирования в рамках той или иной организации. В этом разделе мы рассмотрим основные особенности конфигурации Bacula.

В общем случае для запуска системы Bacula требуется выполнение пяти шагов:

- установка поддерживаемой внешней базы данных и демонов Bacula;
- конфигурирование демонов Bacula;
- установка и конфигурирование демонов управления файлами клиентов;
- запуск демонов Bacula;
- добавление носителей в пул носителей с помощью консоли Bacula.

Мы рассмотрим пример настройки системы состоящей из двух компьютеров: harp и bull. Компьютер harp — клиент, на котором установлен только демон управления файлами. Остальные четыре компонента системы Bacula (демон управления, демон хранения, каталог и консоль) действуют на сервере bull. В качестве устройства хранения используется ленточная библиотека SureStore LTO1. График резервного копирования предусматривает полное еженедельное резервное копирование и ежедневное инкрементное копирование. В качестве каталога используется база данных MySQL.

Установка базы данных и демонов Bacula

Прежде чем можно будет установить систему Bacula, необходимо установить базу данных для хранения ее каталога. В целях упрощения интегрирования с системой Bacula мы рекомендуем использовать базу данных MySQL с применением заданных по умолчанию конфигурационных путей.

Стабильность и надежность — обязательные требования к платформе резервного копирования, поэтому сразу после установки базы данных мы рекомендуем загрузить с Web-сайта Bacula и установить самую последнюю стабильную версию исходного кода. Подробные пошаговые инструкции по установке программы приведены в каталоге `docs` пакета с исходным кодом. Документацию в форматах HTML и PDF можно также найти по адресу `www.bacula.org`. Там же приведены полезные учебники и руководства для разработчиков.

После распаковки исходного кода необходимо выполнить команду

```
./configure --with-mysql --prefix=/etc/bacula
```

а затем команды `make` для компиляции двоичных файлов и `make install`, чтобы закончить инсталляцию.

По завершении установки программы Bacula необходимо создать базу данных MySQL и таблицы данных внутри нее. Убедившись в нормальной работе базы данных MySQL, выполните команду `cd`, чтобы перейти к каталогу установки (`/etc/bacula`) и запустите три сценария интерпретатора, которые были созданы во время выполнения процедуры `make install`. Сценарий `grant_mysql_privileges` предоставляет пользователю Bacula права доступа к базе данных MySQL. Сценарий `create_mysql_database` создает базу данных Bacula, а `make_mysql_tables` заполняет ее необходимыми таблицами.

Конфигурирование демонов Bacula

Прежде чем приступить к подробному описанию процесса установки, необходимо дать определение ряду основных терминов, используемых в системе Bacula.

- "Задания" — это элементарные составляющие активности программы Bacula. Существует две их разновидности: архивирование и восстановление.
- "Пулы" — это группы физических носителей, на которых хранятся задания. В нашем примере конфигурирования мы будем использовать два пула — один для полных архивов и один для инкрементных.
- "Наборы файлов" — это списки логических разделов и файлов. Наборы файлов могут быть явно включены или исключены из заданий архивирования или восстановления.
- Файл "начальной загрузки" — это специальный текстовый файл, созданный программой Bacula, который содержит информацию о файлах, предназначенных для восстановления. Файлы начальной загрузки создаются во время выполнения команды `restore` или во время архивирования, если для задания архивирования определен параметр `Write Bootstrap` (записать файл начальной загрузки).
- "Сообщения" — это информация о состоянии демонов и заданий (фактически журнальные записи), которой обмениваются демоны. Они могут также передаваться по электронной почте или записываться в файлы журналов.

Файлы конфигурации размещаются в каталоге, указанном опцией `--prefix` во время инсталляции (как правило, в каталоге `/etc/bacula`). Следующие разделы содержат описания опции конфигурирования демона управления (`bacula-dir.conf`), демона хранения (`bacula-sd.conf`) и консоли (`bconsole.conf`).

Мы не будем рассматривать все возможные параметры конфигурирования. Вместо этого в начале каждого раздела мы приведем общий обзор файла конфигурации, а затем подробнее остановимся на некоторых параметрах, которые, по нашему мнению, наиболее полезны или трудны для понимания.

Конфигурирование демона управления: файл `bacula-dir.conf`

Файл `bacula-dir.conf` — самый сложный из файлов конфигурации Bacula. Мы рассмотрим десять логических разделов этого файла `Director`, `Catalog`, `JobDefs`, `Job`, `FileSet`, `Schedule`, `Pool`, `Client`, `Storage` и `Messages`. Более подробную информацию о каждом из разделов и его параметрах можно найти в интерактивной документации.

Раздел каждого из ресурсов заключен в фигурные скобки. Во всех файлах конфигурирования программы Bacula комментарии начинаются с символа #.

```
# Пример файла конфигурирования демона управления Bacula, /etc/bacula-dir.conf
Director {
   Name = bull-dir
   DIRport = 9101
   Query File = "/etc/bacula/query.sql"
   Working Directory = "/var/Bacula/working"
   Pid Directory = "/var/run"
   Maximum Concurrent Jobs = 1
   Password = "B@cu1@Lik3s,fRu17"
   Messages = Standard
}
```

Ресурс `Director` фактически служит истоком всех компонентов Bacula. Его параметры определяют имя и основы поведения демона управления. Они определяют коммуникационный порт, через который остальные демоны осуществляют обмен данными с демоном управления, каталог, в котором он хранит свои временные файлы, и количество заданий, которые демон управления может выполнять одновременно.

Параметр `Password` определяет пароль, который программа консоли использует для своей идентификации в демоне управления. Идентичный пароль должен быть определен в ресурсе `Director` файла конфигурации консоли `bconsole.conf`. Хотя в файлах конфигурации пароль записан простым текстом, он никогда не передается по сети.

В данном примере и демон управления и консоль расположены на одном компьютере. Тем не менее, пароль должен присутствовать в обоих файлах конфигурации.

```
# Обобщенная служба каталогов
Catalog {
   Name = ATE
   DBName = bacula;
   user = bacula;
   password = "fRu17,B0wL"
}
```

Раздел Catalog определяет базу данных, в которой Bacula хранит свои рабочие записи. Для простоты мы определили один каталог для всех заданий. Однако Bacula поддерживает применение нескольких каталогов, что позволяет использовать отдельную базу данных для каждой группы клиентов. В настоящее время все базы данных должны размещаться на сервере одной физической базы данных.

Параметр DBName определяет имя базы данных каталога. Ее установка выполняется во время инсталляции в соответствии с параметрами, переданными команде configure. Сценарии заранее построенной базы данных Bacula можно найти в каталоге src/cats пакета с исходным кодом программы.

```
# Определения заданий
Job {
   Name = "harp"
   Level = Full
   Write Bootstrap = "/atrust/admin/backups/bootstraps/harp.bsr"
   Client = harp
   File Set = harp
   Storage = SureStore
   Pool = SSFullPool
   Incremental Backup Pool = SSIncrementalPool
   Schedule = Nightly
}
```

Ресурс Job определяет стандартные параметры конкретного задания архивирования. В общем случае для каждого клиента создается одно определение Job. Большинство параметров совершенно очевидны, но некоторые заслуживают дополнительных пояснений.

Параметр Write Bootstrap указывает программе Bacula о необходимости записи информации начальной загрузки, которая будет использована во время восстановления. Список файлов начальной загрузки содержит перечень файлов и томов, которые требуются для выполнения задания восстановления. Наличие этих файлов начальной загрузки не обязательно, но весьма желательно. Файлы начальной загрузки перезаписываются во время полного архивирования и дописываются во время инкрементного архивирования.

Параметры Client, File Set, Storage, Pool и Schedule представляют собой прямые ссылки на ресурсы, которые определены в последующих разделах файла bacula-dir.conf file.

```
# Определения наборов файлов
FileSet {
   Name = "harp"
   Include {
      Options {
         signature=MD5
         compression=GZIP
      }
      File = "/"
      File = "/boot"
      File = "/usr"
      File = "/usr/local"
      File = "/var"
   }
   Exclude = { /proc /tmp /.journal /.fsck }
}
```

Ресурс FilSet определяет файлы и каталоги, которые должны быть включены или исключены из заданий архивирования. Каждый набор файлов может определять несколько параметров Include и Exclude и индивидуальные параметры Options. По умолчанию программа Bacula рекурсивно архивирует каталоги, но не включает в задание архивирования несколько логических разделов. Все логические разделы, которые нужно архивировать, должны быть указаны в отдельном параметре File.

Программа поддерживает много дополнительных параметров Options, включая регулярные выражения и символы групповой подстановки. Два из них, signature (которому может быть присвоено значение SHA1 или MD5), вычисляющий хеш-значение для каждого архивируемого файла, и compression, который приводит к сжатию данных перед их записью на ленту, заслуживают особого внимания. Параметр signature увеличивает загрузку процессора во время архивирования, но может пригодиться в случае возникновения подозрения о нарушении безопасности.

```
Schedule {
   Name = "Nightly"
   Run = Level=Full Pool=SSFullPool 1st-5th tue at 20:10
   Run = Level=Incremental Pool=SSIncrementalPool wed-mon at 20:10
}
```

Ресурсы Shedule определяют временные таблицы заданий архивирования. Для автоматического выполнения заданий архивирования требуются параметр Name и, по меньшей мере, один параметр Run, но один раздел Shedule может содержать несколько параметров Run. В данном случае полные архивирования выполняются в 20:10 по первым и пятым вторникам каждого месяца, а инкрементные архивирования — еженедельно в 20:10 во все дни с четверга по понедельник.

Спецификации даты и времени — единственные обязательные компоненты параметра Run, но как видно из приведенного примера, можно использовать дополнительные значения параметров. Эти значения заменят стандартные параметры, определенные в спецификации Job. В приведенном примере для каждого задания архивирования мы определили уровень (Level) и пул носителей (Pool).

Полный список ключевых слов спецификаций даты/времени и параметров заданий приведен в интерактивной документации по программе Bacula.

```
# Определения пулов — пул, указанный первым, используется по умолчанию
Pool {
   Name = SSFullPool
   Pool Type = Backup
   Recycle = yes
   Accept Any Volume = yes
}

Pool {
   Name = SSIncrementalPool
   Pool Type = Backup
   Recycle = yes
   Accept Any Volume = yes
}
```

Ресурс Pool группирует носители резервного копирования (обычно ленты) в наборы, используемые конкретными заданиями архивирования. В приведенном примере полные архивирования определены для еженедельного выполнения, а инкрементные — для ежедневного. Поэтому были созданы два ресурса Pool. Поведение этого ресурса опреде-

ляется множеством параметров. Особенно полезны два из них: `Recycle` и `Accept Any Volume`. Если значение параметра `Recycle` установлено равным `yes`, программа Bacula автоматически использует очищенные тома, флаг очистки которых устанавливается, когда для выполнения задания архивирования требуется том, допускающий добавление. Параметр `Accept Any Volume` указывает возможность запись в любой том внутри пула, допускающий добавление. Если значение этого параметра установлено равным `no`, программа Bacula заполняет тома последовательно и не допускает беспорядочное использование томов, которые допускают добавление данных.

```
Client {
  Name = harp
  Address = 192.168.7.2
  FDPort = 9102
  Catalog = ATE
  Password = "Ch@ch1s@Fru17"
  File Retention = 3 months
  Job Retention = 36 months
  AutoPrune = yes
}
```

Ресурсы `Client` определяют компьютеры, файлы которых должны быть архивированы. Для каждого компьютера требуется отдельный ресурс. Параметры `File Retention` и `Job Retention` определяют длительность хранения в каталоге записей файлов и заданий для данного клиента. Если параметр `AutoPrune` установлен, дата истечения срока хранения удаляется из каталога. Эта опция влияет только на записи в каталоге, но не реальные файлы, хранящиеся на архивных лентах.

```
# Определение устройств хранения файлов
Storage {
  Name = SureStore
  Address = bull.atrust.com
  SDPort = 9103
  Password = "Fru1t&V3gg1es"
  Device = SureStoreDevice
  Autochanger = yes
  Media Type = LTO1
}
```

Ресурс `Storage` описывает способ обмена данными с демоном хранения, который управляет устройствами хранения. В нашем примере использована ленточная библиотека SureStore LTO1, подключенная к основному серверу системы Bacula (`bull.atrust.com`). Конфигурация демона хранения определяется отдельным файлом, который рассмотрен в следующем подразделе.

```
Messages {
  Name = Standard
  mailcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
    bacula@atrust.com\" -s \"Bacula: %t %e of %c %l\" %r"
  operatorcommand = "/sbin/bsmtp -h localhost -f \"\(Bacula\)
    bacula@atrust.com\" -s \"Bacula: Intervention needed for %j\" %r"
  mail = Bacula@atrust.com = all, !skipped
  operator = bacula-pager@atrust.com = mount
  console = all, !skipped, !saved
  append = "/var/log/bacula.log" = all, !skipped
}
```

Ресурс Messages указывает программе Bacula способ обработки конкретных типов сообщений, генерируемых каждым из демонов Bacula. Можно определить несколько ресурсов Messages, которые затем будут присвоены конкретным заданиям в соответствующих ресурсах Job.

Этот тип ресурса имеет множество параметров конфигурирования. Полный список соответствующих переменных и команд приведен в интерактивной документации. Приведенная конфигурация отправляет информацию о состоянии демонов и заданий на консоль и в стандартный файл журнала, а также распространяет ее по электронной почте.

Конфигурирование демона хранения: файл bacula-sd.conf

Демоны хранения принимают данные от демонов управления файлов и передают их на реальный носитель хранения (и в обратном направлении при восстановлении). В файле bacula-sd.conf file необходимо определить четыре ресурса: Storage, Device, Messages и Director. Ниже приведен полный пример файла конфигурации.

```
# Файл конфигурации демона хранения, bacula-sd.conf
Storage {
   Name = bull-sd
   SDPort = 9103
   Working Directory = "/var/bacula/working"
   Pid Directory = "/var/run"
}

Device {
   Name = SureStoreDevice
   Media Type = LTO1
   Archive Device = /dev/nst0
   Autochanger = yes
   Changer Device = /dev/sg0
   Changer Command = "/etc/bacula/mtx-changer %c %o %S %a %d"
   AutomaticMount = yes;
   Always Open = yes;
   Removable Media = yes;
   Random Access = no;
}

Messages {
   Name = Standard
   director = bull-dir = all
}

Director {
   Name = bull-dir
   Password = "Fru1t&V3gg1es"
}
```

Файл конфигурации демона хранения должен содержать только один ресурс Storage.

Эти ресурсы сравнительно просты. Они определяют ряд основных рабочих параметров, таких как сетевой порт демона и рабочий каталог, и идентифицируют демон управления, которому демон хранения должен отправлять сообщения.

Ресурс `Device` характеризует реальное устройство архивирования. В данном случае это — накопитель LTO (Linear Tape-Open) с автоматической сменой лент. Допускается определение нескольких ресурсов `Device`. Параметр `Archive Device` определяет имя файла устройства ленточного накопителя. Обратите внимание, что файл устройства `/dev/nst0` соответствует устройству без перемотки, которое требуется практически во всех ситуациях. Устройство автоматической смены ленты обладает собственным файлом устройства. Кроме него в приведенном примере определен параметр `Autochanger`. Параметр `Always Open` указывает о необходимости сохранения устройства в открытом состоянии, если только администратор специально не потребует его размонтирования. Этот параметр экономит время и уберегает ленту от дополнительных нагрузок, поскольку исключает выполнение команд перемотки и позиционирования между заданиями.

Конфигурирование консоли: файл `bconsole.conf`

Программа консоли служит для обмена данными с демоном управления при планировании заданий, проверки их состояния или восстановления данных. Консоль можно запустить из каталога установки, вводя `./bconsole`.

Файл `bconsole.conf` указывает консоли, как следует осуществлять обмен данными с демоном управления Bacula. Параметры в этом файле должны соответствовать определенным в ресурсе `Director` файла конфигурации демона управления (`bacula-dir.conf`), за исключением параметра адреса.

```
# Файл конфигурации консоли, bconsole.conf
Director {
   Name = bull-dir
   DIRport = 9101
   address = bull.atrust.com
   Password = "B@cu1@Lik3s,fRu17"
}
```

Установка и конфигурирование демона управления файлами клиента

Демон управления файлами клиентов архивирования обменивается данными с демоном хранения во время выполнения заданий архивирования и восстановления. Он должен быть установлен и сконфигурирован на каждом компьютере, файлы которого нужно архивировать с помощью программы Bacula.

Для Windows-клиентов заранее построенные двоичные файлы можно загрузить с Web-сайта Bacula. Программа Bacula прекрасно подходит для архивирования файлов данных Windows, но не очень годится для создания абсолютно надежного резервного архива системных файлов Windows. К сожалению, Bacula не поддерживает концепции блокировки открытых файлов Windows, системного реестра Windows или состояния системы, поэтому перед выполнением архивирования с помощью программы Bacula придется выполнить дополнительные действия для сохранения этих данных. Решение этих проблем достаточно подробно описано в интерактивной документации программы Bacula.

В системах UNIX и Linux установку демона можно выполнить, копируя исходное дерево Bacula на каждый компьютер клиента и выполняя команду

```
./configure --enable-client-only --prefix=/etc/bacula
```

Затем необходимо выполнить команды `make` и `make install`.

После того как двоичные файлы установлены, сконфигурируйте демон управления файлами, редактируя файл /etc/bacula/bacula-fd.conf:

```
# Конфигурация демона управления файлами для компьютера harp, bacula-
fd.conf

Director {
  Name = bull-dir
  Password = "Ch@ch1s@Fru17"
}

# "Глобальные" спецификации конфигурации демона управления файлами

FileDaemon {
  Name = harp
  FDport = 9102
  Working Directory = /var/bacula/working
  Pid Directory = /var/run
}

# Отправка всех сообщений, за исключением пропущенных файлов,
# обратно демону управления

Messages {
  Name = Standard
  director = bull-dir = all, !skipped
}
```

Файл конфигурации разбит на три части. Первая состоит из ресурса Director, который указывает демону управления файлами, какой демон управления может планировать задания архивирования из демона управления файлами данного клиента. Ресурс Director содержит также параметр Password, который должен совпадать с паролем, указанным в ресурсе Client собственного файла конфигурации демона управления. Вторая часть файла — ресурс FileDaemon, который определяет имена клиентов и задает порт, прослушиваемый демоном управления файлами на предмет команд от демона управления. Последний компонент — ресурс Messages, который определяет способ обработки локальных сообщений.

Запуск демонов Bacula

После того, как демоны сервера установлены, а клиент сконфигурирован, необходимо запустить демоны с помощью сценария запуска, размещенного в каталоге установки сервера (./bacula start). Эта же команда используется на каждом клиентском компьютере для запуска демона управления файлами. Кроме того, этот сценарий должен быть связан с соответствующими сценариями уровня выполнения в каталогах rc и init.d.

После того как демоны Bacula запущены, программу консоли (bconsole в каталоге установки) можно использовать для проверки их состояния, добавления носителей в пулы и для выполнения заданий архивирования и восстановления. Программу bconsole можно запускать с любого компьютера, если она корректно установлена и сконфигурирована.

```
$ sudo ./bconsole
Password: <введенный пароль>
Connecting to Director bull:9101
1000 OK: bull-dir Version: 1.38.11 (29 June 2006)
Enter a period to cancel a command.
```

Для ознакомления с полным списком поддерживаемых ею команд используйте команду `help` консоли.

Добавление носителей в пулы

Прежде чем можно будет выполнять задания архивирования, необходимо пометить несколько лент и включить их в пулы носителей, определенные в файле конфигурации демона управления. Для этого необходимо использовать команду `label` консоли.

```
* label
Automatically selected Storage: SureStore
Enter new Volume name: 003061L1
Enter slot (0 for none): 14
Defined Pools:
   1: SSFullPool
   2: SSOneOffPool
...
Select the Pool (1-5): 1
Connecting to Storage daemon SureStore at bull.atrust.com:9103
...
Sending label command for Volume "003061L1" Slot 14 ...
...
3001 Mounted Volume: 003061L1
3001 Device /dev/nst0 is mounted with Volume "003061L1"
```

В этом примере ленте, помещенной в слот 14 устройства автоматической смены лент, было присвоено имя `003061L1`, и она была включена в пул `SSFullPool`. Для проверки того, что лента была добавлена в нужный пул и помечена как допускающая добавление данных, следует использовать команду `list media`.

Выполнение архивирования вручную

Чтобы выполнить архивирование вручную, необходимо использовать команду `run` консоли. Эта команда не требует никаких аргументов. При ее выполнении консоль отображает все задания, определенные в файле конфигурации демона управления. Любые параметры команды `run` можно изменять, следуя меню-подобным подсказкам на консоли.

Приведенный ниже пример демонстрирует выполнение вручную полной архивации файлов сервера `harp` с использованием стандартных параметров, указанных в файлах конфигурации.

```
$ sudo ./bconsole
...
Connecting to Director bull:9101
1000 OK: bull-dir Version: 1.34.6 (28 July 2004)
Enter a period to cancel a command.
* run
A job name must be specified.
The defined Job resources are:
       1: harp
       2: RestoreFiles
Select Job resource (1-2): 1
Run Backup job
JobName:   harp
```

```
FileSet:   harp
Level:     Full
Client:    harp
Storage:   SureStore
Pool:      SSFullPool
When:      2006-07-08 13:14:24
Priority:  10
OK to run? (yes/mod/no): yes
Run command submitted.
```

После успешной передачи задания архивирования демону управления его состояние можно отслеживать с помощью команды `status` консоли. Для систематичного получения информации обо всех обновлениях при ее поступлении можно использовать также команду `messages` консоли. В зависимости от настройки ресурсов `Messages` системы подробный итоговый отчет может отправляться также администратору системы Bacula. Приведенный ниже вывод был создан командой `messages` после успешного завершения выполнения задания архивирования. Он содержит много полезной информации.

```
* messages
08-Jul-2006 14:21 bull-dir: Start Backup JobId 5216, Job=harp.2006-07-08_14.21.03
08-Jul-2006 14:23 bull-sd: 3301 Issuing autochanger "loaded drive 0" command.
08-Jul-2006 14:23 bull-sd: 3302 Autochanger "loaded drive 0", result is Slot 6.
08-Jul-2006 14:23 bull-sd: 3303 Issuing autochanger "unload slot 6, drive 0" command.
08-Jul-2006 14:24 bull-sd: 3304 Issuing autochanger "load slot 14, drive 0" command.
08-Jul-2006 14:25 bull-sd: 3305 Autochanger "load slot 14, drive 0", status is OK.
08-Jul-2006 14:25 bull-sd: Volume "003048L1" previously written, moving to end of data.
08-Jul-2006 14:25 bull-sd: Ready to append to end of Volume "003048L1" at file=7.
08-Jul-2006 14:54 bull-dir: Bacula 1.34.6 (28Jul04): 08-Jul-2006 14:54
JobId:                  5216
Job:                    harp.2006-07-08_14.21.03
Backup Level:           Full
Client:                 harp
FileSet:                "harp" 2006-01-06 20:37:06
Start time:             08-Jul-2006 14:21
End time:               08-Jul-2006 14:54
FD Files Written:       176,451
SD Files Written:       176,451
FD Bytes Written:       12,968,821,897
SD Bytes Written:       12,993,427,904
Rate:                   6379.2 KB/sSoftware Compression:None
Volume name(s):         003048L1
Volume Session Id:      263
Volume Session Time:    1149888981
Last Volume Bytes:      13,298,706,752
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:  OK
SD termination status:  OK
Termination:            Backup OK
08-Jul-2006 14:54 bull-dir: Begin pruning Jobs.
08-Jul-2006 14:54 bull-dir: No Jobs found to prune.
08-Jul-2006 14:54 bull-dir: Begin pruning Files.
08-Jul-2006 14:55 bull-dir: Pruned 2,206 Files from 48 Jobs for client harp
from catalog.
08-Jul-2006 14:55 bull-dir: End auto prune.
```

Выполнение задания восстановления

Чтобы восстановить файлы, необходимо запустить консоль и выполнить команду `restore`. Как и команда `run`, команда `restore` управляется из меню. Ее выполнение начинается с вывода информации о том, какие задания должны быть прочитаны для восстановления целевых файлов. Команда `restore` предоставляет несколько методов указания необходимых идентификаторов заданий. После того, как набор заданий выбран, из них можно файлы, которые нужно восстановить.

```
* restore
To select the JobIds, you have the following choices:
       1: List last 20 Jobs run
       2: List Jobs where a given File is saved
       3: Enter list of comma separated JobIds to select
       4: Enter SQL list command
       5: Select the most recent backup for a client
       6: Select backup for a client before a specified time
       7: Enter a list of files to restore
       8: Enter a list of files to restore before a specified time
       9: Cancel
Select item:  (1-9):
```

Вероятно, два наиболее полезные запроса — “Select the most recent backup for a client” (Выберите самую последнюю резервную копию клиента) (№5) и “List Jobs where a given File is saved” (Перечислите задания, содержащие данный файл) (№2). Последняя опция пригодится тем рассеянным пользователям, которые не в состоянии запомнить точное местонахождение случайно удаленного файла. Еще одна полезная команда — опция №4, “Enter SQL list command” (Введите SQL-команду), которая позволяет вводить любой надлежащим образом сформатированный SQL-запрос.

Предположим, что пользователю необходимо скопировать свой сценарий `pw_expire.pl`, восстановленный примерно в апреле 2006 г. Однако он не помнит, какой именно компьютер использовался для разработки в этот период времени. Кроме того, пользователю желательно восстановить файлы в каталог `/tmp` на исходном компьютере. Многих системных администраторов подобных запрос может повергнуть в шоковое состояние, но для администратора Bacula выполнение этой задачи не представляет никакой сложности. (К сожалению, формат вывода результатов поиска, используемый программой Bacula, столь подробен, что нам пришлось максимально уменьшить масштаб выводимой информации.)

```
* restore
To select the JobIds, you have the following choices:
       1: List last 20 Jobs run
       2: List Jobs where a given File is saved
       3: Enter list of comma separated JobIds to select
...
       9: Cancel
Select item:  (1-9): 2
Enter Filename: pw_expire.pl
+-------+-----+-------------------------------+---------------------+--------+---------+----------------+
| JobId |Client| Name                          | StartTime           |JobType |JobFiles |  JobBytes      |
+-------+-----+-------------------------------+---------------------+--------+---------+----------------+
| 4,667 | harp | /home/seidel/pw_expire.pl     | 2006-05-09 23:55:21 | B      |  176,072|  12,664,064,296 |
| 5,220 | bull | /home/seidel/bin/pw_expire.pl | 2006-07-08 20:15:49 | B      |      14 |         248,386 |
| 5,216 | harp | /home/seidel/pw_expire.pl     | 2006-07-08 14:21:05 | B      |  176,451|  12,968,821,897 |
| 4,523 | harp | /home/seidel/pw_expire.pl     | 2006-04-25 23:43:56 | B      |  167,435|  12,539,537,882 |
+-------+-----+-------------------------------+---------------------+--------+---------+----------------+
```

Как видно из списка экземпляров файла pw_expire.pl, созданного программой Bacula, апрельская версия 2006 г. хранилась на компьютере клиента harp и была заархивирована в ходе задания 4523. Затем программа Bacula возвращает нас к меню восстановления, где можно использовать опцию №3 ("Enter list of comma separated JobIds to select" (Введите для выбора список заданий, разделенных запятыми)), чтобы выбрать нужную задачу.

```
Select item: (1-9): 3
Enter JobId(s), comma separated, to restore: 4523
You have selected the following JobId: 4523
Building directory tree for JobId 4523 ...
1 Job inserted into the tree.
You are now entering file selection mode, where you add and
remove files to be restored. All files are initially added.[9]
Enter "done" to leave this mode.

cwd is: /
$ cd /home/seidel
cwd is: /home/seidel
$ dir
...
-rwxr-xr-x 1 seidel atrust   321 2005-10-27 11:25:24 /home/seidel/pw_expire.pl
$ mark pw_expire.pl
1 files marked.
$ done
```

Обратите внимание, что хотя вначале программа отобразила идентификационный номер задания с запятой (4,523), мы ее опустили при повторном вводе. В противном случае программа интерпретировала бы значение 4,523 как два идентификатора заданий: 4 и 523.

После загрузки указанного задания программа переходит в режим интерактивного восстановления, в котором можно просматривать файлы и использовать команду mark для их выбора. По завершении отметки файлов введите команду done, чтобы выйти из интерактивного режима.

```
Bootstrap records written to /var/bacula/working/restore.bsr
The restore job will require the following Volumes:

   003048L1

1 file selected to be restored.

Defined Clients:
   1: bull
   2: harp
Select the Client (1-2): 2
```

Затем программа записывает файл начальной загрузки, который она будет использовать для выполнения восстановления, отображает имена необходимых ей ленточных томов и предлагает выбрать клиентский компьютер, на котором нужно восстановить файлы. В данном примере восстановление было выполнено на исходном компьютере harp.

[9] Стоп, стоп, стоп! Если продолжить этап восстановления без выбора каких-либо файлов, программа сообщит, что никакие файлы не были выбраны.

```
Run Restore job
JobName:        RestoreFiles
Bootstrap:      /var/Bacula/working/restore.bsr
Where:          /scratch02/restore/
Replace:        always
FileSet:        BullHomes
Client:         harp
Storage:        SureStore
When:           2006-07-08 16:45:20
Priority:       10
OK to run? (yes/mod/no): mod
```

Для данного конкретного задания параметры, заданные по умолчанию были изменены. В частности, правильное имя набора файлов — `harp`, которое представляет набор файлов, определенный по умолчанию в файле конфигурации демона управления. Кроме того, в соответствии с пожеланием пользователя мы изменили место назначения этого восстановления на `/tmp`.

```
Select parameter to modify (1-11): 9
Please enter path prefix for restore (/ for none): /tmp
Run Restore job
JobName:        RestoreFiles
Bootstrap:      /var/Bacula/working/restore.bsr
Where:          /tmp
Replace:        never
FileSet:        harp
Client:         harp
Storage:        SureStore
When:           2006-07-08 16:45:20
Priority:       10
OK to run? (yes/mod/no): yes
Run command submitted.
Restore command done.
```

После внесения изменений мы передаем задание демону управления, которые его выполняет. Затем можно воспользоваться командой `messages` для просмотра журнала выполнения задания.

```
* messages
08-Jul-2006 17:06 bull-dir: Start Restore Job RestoreFiles.2006-07-08_17.06.02
08-Jul-2006 17:06 bull-sd: Ready to read from volume "003048L1" on device /dev/nst0.
08-Jul-2006 17:06 bull-sd: Forward spacing to file:block 11:0.
08-Jul-2006 17:08 bull-sd: End of Volume at file 11 on device /dev/nst0, Volume
    "003048L1"
08-Jul-2006 17:08 bull-sd: End of all volumes.
08-Jul-2006 17:08 harp: -rwxr-xr-x   1 seidel   atrust      321 2005-10-27 11:25:24
    /tmp/home/seidel/pw_expire.pl
08-Jul-2006 17:08 bull-dir: Bacula 1.34.6 (28Jul04): 08-Jul-2006 17:08
JobId:                  5217
Job:                    RestoreFiles.2006-07-08_17.06.02
Client:                 harp
Start time:             08-Jul-2006 17:06
End time:               08-Jul-2006 17:08
Files Expected:         1
Files Restored:         1
Bytes Restored:         321
```

```
Rate:                    0.0 KB/s
FD Errors:               0
FD termination status:   OK
SD termination status:   OK
Termination:             Restore OK

08-Jul-2006 17:08 bull-dir: Begin pruning Jobs.
08-Jul-2006 17:08 bull-dir: No Jobs found to prune.
08-Jul-2006 17:08 bull-dir: Begin pruning Files.
08-Jul-2006 17:08 bull-dir: No Files found to prune.
08-Jul-2006 17:08 bull-dir: End auto prune.
```

Мониторинг и отладка конфигураций программы Bacula

Команду `status` консоли можно использовать для запроса различной информации демонов Bacula. В следующем примере отображается информация о демоне управления. Вывод содержит информацию о запланированных, выполняющихся и прерванных заданиях.

```
* status dir
bull-dir Version: 1.34.6 (28 July 2004) i686-redhat-linux-gnu redhat
Enterprise release
Daemon started 09-Jun-06 15:36, 269 Jobs run since started.

Scheduled Jobs:
Level               Type           Scheduled           Name     Volume
===================================================================================
Incremental         Backup         08-Jul-06 20:10     harp     003005L1

Running Jobs:
JobId   Level      Name                                Status
======================================================================
5216    Full       harp.2006-07-08_14.21.03            is running

Terminated Jobs:
JobId   Level      Files    Bytes          Status   Finished          Name
====================================================================
5205    Incr       204      898,066,578    OK       06-Jul-06 20:36   harp
```

Задания Bacula создают отчет, который направляется согласно ресурсу `Message` в файле конфигурации демона управления. Отчет содержит основную информацию об использованных томах, размере и количестве заархивированных файлов, и о любых возникших ошибках. Как правило, он предоставляет достаточный объем информации для решения любых не очень серьезных проблем.

Две наиболее часто встречающиеся проблемы — невозможность запуска демонов управления файлами клиентов и неспособность демонов хранения обнаружить какие-либо ленточные тома, доступные для добавления данных. В приведенном ниже примере демон управления сообщает, что задание архивирования было прервано с фатальной ошибкой вследствие невозможности осуществления обмена данными с демоном управления файлами компьютера `harp`. Сообщение об этой ошибке может многократно повторяться в конце итогового отчета.

```
bull-dir: Start Backup JobId 5215, Job=harp.2006-07-08_13.19.49
bull-dir: harp.2006-07-08_13.19.49 Error: Bacula 1.34.6 (28Jul04): 08-Jul-2006 13:49
```

```
JobId:                  5215
Job:                    harp.2006-07-08_13.19.49
Backup Level:           Full
Client:                 harp
FileSet:                "harp" 2006-01-06 20:37:06
Start time:             08-Jul-2006 13:19
End time:               08-Jul-2006 13:49
FD Files Written:       0
SD Files Written:       0
FD Bytes Written:       0
SD Bytes Written:       0
Rate:                   0.0 KB/s
Software Compression:   None
Volume name(s):
Volume Session Id:      262
Volume Session Time:    1149888981
Last Volume Bytes:      0
Non-fatal FD errors:    0
SD Errors:              0
FD termination status:
SD termination status:  Waiting on FD
Termination:            *** Backup Error ***
bull-dir: harp.2006-07-08_13.19.49 Warning: bnet.c:666 Could not connect to
      File daemon on 192.168.7.2:9102. ERR=Connection refused Retrying ...
bull-dir: harp.2006-07-08_13.19.49 Fatal error: bnet.c:672 Unable to connect to
      File daemon on 192.168.7.2:9102. ERR=Connection refused
```

В следующем примере демон хранения сообщает, что для запрошенного задания архивирования отсутствуют доступные ленточные тома соответствующего пула. Проблему можно устранить либо добавив в пул новый том, либо очистив и повторно используя существующий том. Задание не нужно запускать снова — программа Bacula должна продолжить его выполнение, если только оно не будет прервано явно.

```
bull-sd: Job baikal.2006-07-04_20.10.06 waiting. Cannot find any appendable volumes.
Please use the "label"  command to create a new Volume for:
    Storage:        SureStoreDevice
    Media type:     LTO1
    Pool:           SSFullPool
```

Чтобы ознакомиться с более подробной информацией о действиях, выполняемых демонами, их можно вынудить отсылать отладочную информацию на консоль, добавив опцию `-d`*`nnn`* к команде запуска. Например, можно использовать команду

```
$ sudo ./bacula start -d100
```

Значение *nnn* представляет уровень отладки. Типичные значения лежат в диапазоне от 50 до 200. Чем выше это значение, тем больший объем информации отображает команда. Отладку можно также активизировать из консоли с помощью команды `setdebug`.

Альтернативы программе Bacula

В Internet доступно несколько других бесплатных или условно бесплатных пакетов резервного копирования. Следующие пакеты заслуживают особого внимания, хотя они все еще находятся в состоянии активной разработки.

- Amanda: очень популярная и проверенная система, которая выполняет архивирование файлов систем UNIX и Linux на одном ленточном устройстве. См. `www.amanda.org`.
- Mondo Rescue: утилита, которая выполняет архивирование файлов систем Linux на CD-R, DVD-R, ленту или жесткий диск. Этот пакет особенно удобен, если необходимо выполнить восстановление в системе, лишенной какого-либо программного обеспечения. Подробнее о нем можно узнать по адресу `www.mondorescue.org`.
- `rsync`: бесплатная утилита, входящая в состав многих пакетов Linux. Она может выполнять синхронизацию файлов с одного компьютера на другой и ее можно использовать совместно с SSH для безопасной передачи данных по Internet. Дополнительная информация о программе `rsync` приведена в разделе 17.3.
- `star`: более производительная реализация программы `tar`. Эта программа входит в состав всех рассматриваемых нами примеров дистрибутивов.

9.9. Коммерческие системы резервного копирования

Нам бы хотелось, чтобы Linux была единственной операционной системой в мире, но, к сожалению, это не так. Анализируя различные коммерческие системы резервного копирования, необходимо учитывать их способность взаимодействовать с другими операционными системами, которые используются в организации. Большинство современных продуктов позволяет включать рабочие станции UNIX, Windows и Macintosh в схему резервного копирования, осуществляемого в Linux. Следует также учитывать наличие массивов хранилищ и файловых серверов, работающих не под управлением Linux.

Защищать от сбоев нужно также портативные компьютеры пользователей и другие компьютеры, которые не имеют постоянного подключения к сети. Коммерческая программа должна правильно обрабатывать ситуации, когда архивируются идентичные файлы с каждого портативного компьютера. В конце концов, сколько резервных копий файла `command.com` вам нужно?

Поскольку система Bacula идеально подходит для наших потребностей, мы не приобрели богатого опыта работы с другими коммерческими пакетами. Мы опросили нескольких знакомых администраторов из коммерческих организаций на предмет их впечатлений от систем, которыми они пользуются. Ниже приведены их комментарии.

ADSM/TSM

Пакет ADSM был разработан компанией IBM и впоследствии приобретен компанией Tivoli. Сегодня он носит название TSM (Tivoli Storage Manager — диспетчер систем хранения компании Tivoli), хотя права собственности снова отошли к IBM, и представляет собой систему управления данными, поддерживающую также резервное копирование. Дополнительную информацию можно найти на Web-сайте `www-306.ibm.com/software/tivoli`.

Достоинства:

- принадлежность компании IBM;
- удачная ценовая политика;
- низкий процент сбоев;

- использование дисковых кэш-буферов, что удобно при работе с низкоскоростными клиентами;
- поддержка клиентов Windows;
- великолепная документация (за отдельную плату).

Недостатки:

- плохой графический интерфейс;
- каждые два файла занимают 1 Кбайт в базе данных;
- сложность системы постоянно возрастает.

Veritas

В 2005 г. произошло слияние компаний Veritas и Symantec. Объединенная компания продает системы резервного копирования для различных платформ. Информацию о них можно получить на Web-сайте `www.symantec.com`.

Достоинства:

- неплохой графический интерфейс;
- прямое взаимодействие с файловыми серверами Network Appliance;
- инсталляция по методу принудительной рассылки в Linux;
- возможность записи лент в формате GNU-версии команды `tar`;
- централизованная база данных, но поддерживается и распределенное резервное копирование.

Недостатки:

- ряд программных ошибок;
- неудачная ценовая политика.

Прочие программы

Уже упоминавшийся Кертис Престон, автор книги по резервному копированию, вышедшей в издательстве O'Reilly, ведет очень полезную Web-страницу `www.backupcentral.com`, посвященную данной тематике (программы зеркального дублирования дисков, работы с файловыми системами, удаленного резервного копирования, внешнего хранения данных и др.). Не совсем понятно, обновляется ли какая-либо часть этой информации, но и существующие материалы весьма полезны.

9.10. Рекомендуемая литература

Preston, W. Curtis. *Backup & Recovery*. Sebastopol, CA: O'Reilly Media, 2006.

9.11. Упражнения

9.1 Проанализируйте процедуру резервного копирования, применяемую в вашей компании или организации. На каком компьютере (или компьютерах, если их несколько) выполняется резервное копирование? Какие типы накопителей используются? Где хранятся ленты? Можно ли улучшить существующую схему?

9.2. Какие действия необходимо предпринять для восстановления файлов в системе Bacula? Как найти нужную ленту?

9.3. Укажите действия, которые необходимо осуществить для выполнения трех запрошенных операций восстановления при наличии следующего вывода команд `df` и `/etc/dumpdates`. Аргументируйте свое решение. Примите, что запрос на восстановления датируется 18 января.

Вот результаты выполнения команды `df` на сайте `khaya.cs.colorado.edu`:

```
/dev/hda8        256194     81103    161863   33%  /
/dev/hda1         21929      4918     15879   24%  /boot
/dev/hda6       3571696     24336   3365924    1%  /local
/dev/hda10       131734      5797    119135    5%  /tmp
/dev/hda5       1815580   1113348    610004   65%  /usr
/dev/hda7        256194     17013    225953    7%  /var
```

А вот содержимое файла `/etc/dumpdates` на `khaya.cs.colorado.edu`:

```
/dev/hda8                2 Tue Jan   17   22:59:23 2006
/dev/hda6                3 Tue Jan   17   22:51:51 2006
/dev/hda7                3 Tue Jan   17   22:50:24 2006
/dev/hda5                9 Tue Jan   17   22:46:25 2006
/dev/hda5                1 Thu Jan   12   22:45:42 2006
/dev/hda7                0 Thu Jan   12   23:14:47 2006
/dev/hda6                1 Thu Jan   12   23:14:32 2006
/dev/hda8                1 Thu Jan   12   23:14:17 2006
/dev/hda6                0 Tue Jan   10   22:47:31 2006
/dev/hda1                1 Sun Jan    8   22:16:05 2006
/dev/hda7                1 Sat Jan    7   22:08:09 2006
/dev/hda1                4 Tue Jan    3   22:51:53 2006
/dev/hda7                2 Sat Dec   24   22:53:52 2005
/dev/hda5                0 Thu Nov    3   22:46:21 2005
/dev/hda1                0 Wed Sep   21   22:46:29 2005
/dev/hda8                0 Wed Aug   24   23:01:24 2005
/dev/hda1                3 Fri Jul   29   22:52:20 2005
/dev/hda6                2 Fri Jul   29   23:01:32 2005
```

а) “Пожалуйста, восстановите весь мой домашний каталог (`/usr/home/clements`) по состоянию на несколько дней до настоящего момента. Похоже, что я потерял всю кодовую базу своего проекта.”

б) “Я случайно выполнил команду `sudo rm -rf /*` на своем компьютере `khaya`. Не могли бы Вы восстановить все файловые системы из последних резервных копий?”

в) “Все мои MP3-файлы, которые я скачивал из BitTorrent в течение более месяца, пропали. Они хранились в каталоге `/tmp/mp3/`. Не могли бы вы восстановить их?”

☆ 9.4. Разработайте схему резервного копирования для описанных ниже сценариев. Предполагается, что каждый компьютер оснащен жестким диском емкостью 100 Гбайт и домашние каталоги пользователей хранятся локально. Выберите устройство резервного копирования, оптимальное по соотношению стоимости и трудоемкости эксплуатации. Аргументируйте свой выбор.

а) Научно-исследовательский центр с 50 компьютерами. На каждом компьютере хранится множество важных и часто изменяемых данных.

б) Небольшая компания, занимающаяся выпуском программного обеспечения и располагающая десятью компьютерами. Исходные коды хранятся на сервере, емкость жесткого диска которого составляет 500 Гбайт. Разрабатываемые исходные коды постоянно меняются в течение дня, а домашние каталоги пользователей меняются редко. Стоимость системы резервного копирования не играет особой роли, зато крайне важна безопасность данных.

в) Домашняя сеть с двумя компьютерами. Здесь основной фактор — это цена.

☆ 9.5. Разработайте стратегию восстановления архивов для каждого из трех вышеперечисленных сценариев.

☆ 9.6. Опишите действия, которые нужно было бы предпринять для безопасного выполнения команды `rdump` по безопасному туннелю SSH.

☆☆ 9.7. Напишите операторы конфигурирования Bacula, которые реализуют план архивирования, разработанный для упражнения 9.4.

Book_LinuxAdmHand.indb 250 17.04.2007 13:54:59

Глава 10

Система Syslog и журнальные файлы

Системные демоны, ядро, утилиты — все они генерируют журнальные данные, которые регистрируются и попадают на далеко не безразмерные диски. "Срок полезной службы" большинства данных ограничен, поэтому их приходится группировать, сжимать, архивировать и, наконец, удалять.

10.1. Методы обработки журнальных файлов

Принципы работы с журнальными файлами могут быть самыми разными. Вот некоторые общепринятые схемы:

- немедленное уничтожение всех журнальных файлов;
- сброс журнальных файлов в исходное состояние через определенные промежутки времени;
- ротация журнальных файлов (данные хранятся в течение фиксированного периода времени);
- сжатие журнальных файлов и архивирование их на ленту или постоянный носитель другого типа.

Правильный выбор стратегии зависит от того, сколько имеется свободного дискового пространства и какое внимание уделяется вопросам безопасности данных. Даже те пользователи, которые не испытывают недостатка в дисковом пространстве, должны учитывать вероятность неприемлемого разрастания журнальных файлов.

Независимо от выбранной схемы процедуру управления журнальными файлами необходимо автоматизировать с помощью демона `cron`. Общие сведения об этом демоне были приведены в главе 8.

Уничтожение журнальных файлов

Не рекомендуется удалять всю журнальную информацию. Организации, сталкивающиеся с проблемами безопасности, всегда могут обнаружить в журнальных файлах важные доказательства имевших место взломов. Кроме того, журнальные файлы весьма полезны как способ привлечения внимания администратора к проблемам, возникающим в аппаратных и программных средствах. При наличии достаточного объема дискового пространства журнальные данные обычно хранят как минимум месяц. На практике именно столько времени может уйти на выяснение того, была ли система атакована хакером, и просмотр журнальных файлов. Если же требуется вернуться в более отдаленное прошлое, можно восстановить старые файлы с архивных лент.

Некоторые администраторы позволяют журнальным файлам разрастаться до тех пор, пока их размер не начнет вызывать раздражение, после чего файлы обнуляются. Это, конечно, лучше, чем вообще не хранить регистрационные данные, но такой метод не гарантирует, что важные элементы журнального файла сохранятся в течение определенного промежутка времени. Кроме того, средний объем занимаемого дискового пространства при таком подходе может оказаться выше, чем в других методах журнальной регистрации.

В некоторых организациях исходят из того, что журнальные файлы нужны лишь на крайний случай и не дают никаких реальных преимуществ. При таком подходе журнальные файлы могут храниться в течение нескольких недель, но они никогда не переносятся на постоянный носитель. За примерами не нужно далеко ходить: лишь недавно компании Microsoft было предъявлено несколько судебных исков по поводу неоправданно деструктивной политики в отношении журнальных файлов и сообщений электронной почты. Истцы обвиняют компанию Microsoft в том, что применяемая ею политика хранения данных ведет к уничтожению фактических данных, несмотря на то, что удалению (или, по крайней мере, применению политики удаления данных) предшествуют конкретные юридические действия. К сожалению, пока еще рано говорить о том, как суды отреагируют на эти иски[1]. Недавно в США закон Сарбейнса-Оксли, 2002 определил новые требования к хранению учетных данных (см. раздел 30.12).

Ротация журнальных файлов

Часто (хотя и не всегда) журнальную информацию за каждую неделю или месяц хранят в отдельном файле. Эти файлы за конкретный период остаются в системе в течение определенного времени, а затем удаляются. В нашей системе мы выделяем под журнальные файлы целый раздел диска (`/var/log`) на центральном хосте.

В конце принятого ротационного периода специальный сценарий или служебная программа переименовывает файлы, в результате чего более старые данные сдвигаются в конец цепочки. Если, например, журнальный файл называется `logfile`, то резерв-

[1] Следует отметить, что удаление важных сообщений электронной почты или журнальных файлов сразу после возникновения угрозы судебного иска (или, хуже того, во время судебного процесса) совершенно недопустимо. Лишь явная, применяемая постоянно политика удаления устаревших данных может быть признана законной.

ные копии могут носить имя `logfile.1`, `logfile.2` и т.д. Если в системе хранится 8-недельный комплект данных, а ротация выполняется еженедельно, то последняя копия будет называться `logfile.8`, а файла `logfile.9` уже не будет. Еженедельно данные в файле `logfile.8` пропадают, так как на их место записываются данные из `logfile.7`.

С целью удлинения срока хранения можно использовать сжатие данных, хотя это и ведет к некоторому усложнению работы. Для просмотра сжатых файлов без их постоянной распаковки можно использовать утилиту `zgrep`.

Предположим, некий журнальный файл обновляется еженедельно, а его содержимое нужно хранить в течение трех дней. Соответствующий цикл ротации будет реализован с помощью такого сценария:

```
#!/bin/sh
cd /var/log
mv logfile.2 logfile.3
mv logfile.1 logfile.2
mv logfile logfile.1
cat /dev/null > logfile
chmod 600 logfile
```

Для некоторых журнальных файлов большое значение имеет информация о владельцах. Можно запускать сценарий ротации средствами демона `cron` от имени владельца журнальных файлов, а не пользователя root, или же добавить в сценарий команду `chown`.

В большинстве дистрибутивов Linux (включая и наши демонстрационные дистрибутивы) имеется очень удобная утилита ротации журналов `logrotate`, описываемая в разделе 10.3. Пользоваться ею гораздо проще (и надежнее), чем писать собственные сценарии. Если этой утилиты нет в системе, ее стоит инсталлировать.

Иногда журнальные файлы классифицируют по датам, а не порядковым номерам, например `logfile.tues` или `logfile.2005.04.01`. Такую схему ротации немного труднее реализовать, но затраты окупятся, если необходимость обращаться к старым журнальным файлам возникает довольно часто. Эту схему проще реализовать с помощью сценария Perl, а не сценария интерпретатора `sh`. Вот самая простая реализация:

```
mv logfile logfile.`date +%Y.%m.%d`
```

Теперь с помощью команды `ls` можно выводить список журнальных файлов в хронологическом порядке. (Опция `-t` команды `ls` также вынуждает ее сортировать файлы в хронологическом порядке по времени их изменения в любом каталоге, но гораздо удобнее, когда упорядочение файлов выполняется автоматически без дополнительного запроса.)

Некоторые демоны все время держат свои журнальные файлы открытыми. Из-за особенностей работы файловой системы приведенный выше сценарий нельзя использовать с такими демонами: журнальные данные будут направляться не во вновь созданный журнальный файл, а по-прежнему — в файл `logfile.1`. Это объясняется тем, что активная ссылка (открытый дескриптор) на исходный файл сохраняется даже после переименования файла. Для того чтобы инсталлировать новый журнальный файл, нужно либо послать демону специальный сигнал, либо уничтожить и перезапустить демон. Каждая программа выполняет обработку журнальных файлов по-своему. Для ознакомления с процедурами, которые требуются в каждом конкретном случае, обратитесь к соответствующей главе этой книги (или имеющейся в вашем распоряжении документации).

Вот текст несколько более подробного сценария, в котором используется сжатие журнальных файлов и посылается сигнал демону:

```
#!/bin/sh
cd /var/log
mv logfile.2.gz logfile.3.gz
mv logfile.1.gz logfile.2.gz
mv logfile logfile.1
cat /dev/null > logfile
chmod 600 logfile
kill -сигнал идентификатор_процесса
gzip logfile.1
```

В поле *сигнал* указывается код сигнала для программы, которая записывает данные в журнальный файл; поле *идентификатор_процесса* — это идентификатор ее процесса. Сигнал жестко задается в сценарии, а вот идентификатор процесса нужно определять динамически: либо путем чтения специального файла (например, `/var/run/syslogd.pid`; описан далее в главе), либо с помощью команды `killall`. Последняя представляет собой разновидность команды `kill`, которая самостоятельно ищет идентификатор указанного процесса.

Например, команда

```
killall -e -HUP syslogd
```

эквивалентна команде

```
kill -HUP `cat /var/run/syslogd.pid`
```

Архивирование журнальных файлов

📖 Подробно о резервном копировании рассказывалось в главе 9.

Журнальные файлы всегда следует включать в процедуру резервного копирования, если только не требуется явно предотвратить уничтожение следов выполненных действий. Они содержат информацию, без которой не обойтись при расследовании инцидентов, связанных с безопасностью, поэтому такие файлы следует архивировать настолько часто, насколько позволяет применяемая схема резервного копирования. Вследствие регулярных изменений журнальные файлы могут составлять основную часть инкрементных архивов. Этот факт следует учитывать при разработке схем журнальной регистрации и резервного копирования.

Журнальные файлы можно помещать в обычные системные архивы либо на отдельные ленты. Второй вариант более обременителен, но он требует меньше усилий по составлению документации и дает возможность циклически использовать архивные ленты, не затрагивая резервные копии журнальных файлов. В подобном случае мы рекомендуем пользоваться команду `tar` и написать сценарий автоматизации резервного копирования.

10.2. Журнальные файлы Linux

Традиционные UNIX-системы часто критикуют за несогласованный и даже несколько странный подход к управлению журнальной регистрацией. К счастью, Linux в этом плане — гораздо более интеллектуальная система, хотя в каждом дистрибутиве

журнальные файлы и названы, и распределяются по каталогам по-разному. В основном Linux-приложения записывают журнальную информацию в файлы, находящиеся в каталоге `/var/log`. В некоторых дистрибутивах для этой цели используется также каталог `/var/adm`.

📖 Формат файла `syslog.conf` описан в разделе 10.4.

В настоящее время большинство программ посылает свои журнальные данные системе Syslog (рассматривается далее), которая выполняет контрольно-распределительные функции. Стандартная конфигурация этой системы такова, что основная часть сообщений попадает как раз в файлы каталога `/var/log`. Точные места размещения сообщений можно выяснить, просмотрев конфигурационный файл `/etc/syslog.conf`.

В табл. 10.1 представлена информация о наиболее часто используемых журнальных файлах демонстрационных дистрибутивов. Указаны, в частности, следующие сведения:

- журнальные файлы, подлежащие архивированию или другой обработке;
- программа, создающая каждый из этих файлов;
- информация о том, где задается имя файла;
- периодичность удаления устаревшей информации, которая считается приемлемой;
- дистрибутивы (из числа демонстрационных), в которых используется каждый из этих файлов;
- описание содержимого файлов.

Имена файлов даны относительно каталога `/var/log`, если не указано иное.

Буква в столбце "Где" указывает на то, где определяется имя журнального файла: S — в системе Syslog; Ф — в конфигурационном файле соответствующей программы; В — в ядре. В столбце "Частота" предложена периодичность чистки журнальных файлов. В столбце "Дистрибутив" указаны дистрибутивы, в которых используется данный журнальный файл.

Журнальные файлы обычно принадлежат пользователю root, хотя соглашения о владельцах и режимах доступа к этим файлам неодинаковы в разных дистрибутивах. По нашему мнению, большинство журнальных файлов должно иметь код доступа 600 (чтение и запись разрешены только владельцу), поскольку их содержимое может представлять потенциальный интерес для хакеров. Если же пользователи системы выполняют сравнительно сложные задачи, возможность просмотра журнальных файлов обеспечит им определенные преимущества. В этом случае целесообразно до некоторой степени снизить ограничения доступа к определенному поднабору журнальных файлов.

Еще один разумный компромисс — присвоить права владения журнальными файлами специально созданной группе и открыть доступ для чтения этих файлов только членам этой группы. В нее можно включить локальных системных администраторов, предоставив им права просмотра журнальных файлов без необходимости использования команды `sudo`. Такое конфигурирование особенно полезно, если в организации работают младшие системные администраторы, которые не должны иметь полные привилегии `sudo`.

По крайней мере, файлы `secure`, `auth.log` и `sudo.log` точно должны быть недоступны для просмотра. Правом записи в любой журнальный файл должен обладать только его владелец.

Таблица 10.1. Журнальные файлы

Файл	Программа	Где[а]	Частота[а]	Дистрибутив[а]	Содержимое/назначение
auth.log	su и др.[б]	S	M	DU	Авторизационные сообщения
apache2/*	httpd версии 2	Ф	Д	SDU	Журналы HTTP-сервера Apache (версия 2)
boot.log	Сценарии запуска системы	Ф[в]	M	RF	Выходная информация сценариев запуска
boot.msg	Ядро	B	–	S	Образ буфера сообщений ядра
cron	cron	S	H	RF	Сведения о выполнении и об ошибках демона cron
cups/*	CUPS	C	H	все	Сообщения, связанные с печатью (CUPS)
daemon.log	Различные	S	H	DU	Все сообщения средств демона
debug	Различные	S	Д	DU	Отладочные сообщения
dmesg	Ядро	B	–	RFDU	Образ буфера сообщений ядра
dpkg.log	dpkg	Ф	M	DU	Журнал управления пакетом
faillog	login	B	H	SDU	Сообщения о неудачных попытках регистрации в системе
httpd/logs/*	httpd	Ф	Д	RF	Журналы HTTP-сервера Apache (в каталоге /etc)
kern.log	Ядро	S	H	DU	Все сообщения средств ядра
lastlog	login	B	–	все	Время последней регистрации в системе каждого пользователя (бинарный файл)
mail*	Связанные с электронной почтой	S	H	все	Все сообщения средств электронной почты
messages	Различные	S	H	все	Часто это основной системный журнальный файл
rpmpkgs	cron.daily/rpm	B	Д	RF	Список установленных RPM-пакетов
samba/*	smbd и др.	C	H	–	Samba (совместное использование файлов в системах Windows/CIFS)
secure	sshd, sudo и т.п.	S	M	RF	Конфиденциальные авторизационные сообщения
syslog	Различные	S	H	DU	Часто это основной системный журнальный файл
warn	Различные	S	H	S	Все сообщения уровня предупреждений/ошибок
wtmp	login	B	M	все	Сообщения о регистрации в системе (бинарный файл)
Xorg.*n*.log	Xorg	Ф	H	RFS	Сообщения об ошибках сервера X Windows
yum.log	yum	Ф	M	RF	Журнал управления пакетом

[а] Столбец "Где" (способ задания имени файла): S — Syslog, B — встроенное имя, Ф — конфигурационный файл. Столбец "Частота" (рекомендуемая частота проверки): Д — ежедневно, H — еженедельно, M — ежемесячно. Столбец "Дистрибутив": R — Red Hat Enterprise Linux, F- Fedora, D — Debian, S — SuSE, U — Ubuntu.

[б] Команды passwd, login и shutdown тоже записывают информацию в этот файл. В системах Red Hat и Fedora для этого используется каталог /var/adm.

[в] В действительности сообщения направляются в систему Syslog, но название средства и уровень важности задаются в файле /etc/initlog.conf.

Следует отметить, что многие журнальные файлы из числа перечисленных в табл. 10.1 контролируются системой Syslog, но стандартная ее конфигурация сильно зависит от дистрибутива. Если бы файл `/etc/syslog.conf` был более согласованным, управление журнальными файлами велось бы примерно одинаково во всех дистрибутивах Linux.

Специальные журнальные файлы

Большинство журнальных файлов — это обычные текстовые файлы, в которые записываются сообщения о "важных" событиях. Но некоторые файлы из числа перечисленных в табл. 10.1 имеют совершенно другое назначение.

Файл `/var/log/wtmp` содержит записи о том, когда пользователи входили в систему и выходили из нее, а также когда система выключалась или перезагружалась. Это довольно простой файл (новые записи просто добавляются в конец), тем не менее, он хранится в бинарном виде. Расшифровать содержимое файла можно с помощью команды `last`. Несмотря на не совсем обычный формат, файл `wtmp` должен подвергаться ротации и усечению, как и любой другой журнальный файл, поскольку имеет тенденцию к чрезмерному разрастанию.

🕮 Подробнее о разреженных файлах см. сноску в разделе 9.3.

В файле `/var/log/lastlog` регистрируется время последнего входа в систему каждого пользователя. Это разреженный бинарный файл, записи которого индексируются по идентификатору пользователя. Размер файла будет меньше, если назначать идентификаторы по порядку, хотя вряд ли об этом стоит сильно беспокоиться. Файл `lastlog` не должен участвовать в цикле ротации, поскольку его размер в большинстве случаев остается неизменным.

Журнальная регистрация на уровне ядра и на этапе начальной загрузки

Ядро и сценарии запуска системы представляют собой немалую проблему с точки зрения журнальной регистрации. В первом случае проблема заключается в том, чтобы средства регистрации событий, происходящих на этапе начальной загрузки и в ядре, не зависели от типа файловой системы и ее организации. Во втором случае трудность состоит в последовательной регистрации событий, происходящих в сценариях, а не в системных демонах или других программах — их сообщения должны регистрироваться на другом уровне. Необходимо также избежать ненужного дублирования записей в сценариях и переадресации их вывода.

Первая из описанных проблем решается путем создания внутреннего буфера ограниченного размера, куда ядро заносит свои журнальные сообщения. Этот буфер достаточно велик для того, чтобы вместить сообщения обо всех действиях, предпринимаемых ядром на этапе начальной загрузки. Когда система полностью загрузится, пользовательский процесс сможет получить доступ к журнальному буферу ядра и скопировать его содержимое в нужное место. Обычно для этого запускается команда `dmesg`, выводные данные которой перенаправляются в файл `/var/log/dmesg` (в RHEL, Fedora, Debian и Ubuntu) или `/var/log/boot.msg` (в SUSE). Именно там лучше всего искать информацию о последнем цикле начальной загрузки системы.

Сообщения, генерируемые ядром в процессе работы, обрабатываются демоном `klogd`. С функциональной точки зрения он расширяет возможности команды `dmesg`. Демон может или просто создать образ журнального буфера ядра и завершить работу, или динамически извлекать сообщения из буфера по мере их появления, записывая их в файл или передавая системе Syslog. По умолчанию демон работает во втором режиме. Его сообщения обрабатываются системой Syslog в соответствии с установками для средства "kern" (обычно они записываются в каталог `/var/log/messages`).

В наших демонстрационных дистрибутивах сценарии запуска не используют флаг `-c` команды `dmesg` при создании начального образа журнального буфера, вследствие чего содержимое буфера читается, но не сбрасывается. Когда демон `klogd` запускается, он находит в буфере те же сообщения, которые прочла команда `dmesg`, и передает их системе Syslog. По этой причине некоторые сообщения появляются не только в файле `dmesg` или `boot.msg`, но и в файле `/var/log/messages`.

Другой важный момент, связанный с журнальной регистрацией на уровне ядра, касается правильного управления системной консолью. Важно, чтобы по ходу загрузки системы вся выводимая информация попадала на консоль. С другой стороны, когда система загрузилась и начала выполняться, консольные сообщения больше раздражают, нежели помогают, особенно если консоль используется для регистрации в системе.

Команда `dmesg` и демон `klogd` позволяют указывать в командной строке уровень важности консольных сообщений ядра, например:

```
$ sudo dmesg -n 2
```

Сообщения уровня 7 наиболее информативны и включают отладочную информацию. К первому уровню относятся только сообщения о фатальных ошибках системы (чем ниже уровень, тем выше важность сообщений). Все сообщения, выдаваемые ядром, продолжают поступать в журнальный буфер (а также в систему Syslog) независимо от того, направляются они на консоль или нет.

В каталог `/proc/sys` ядро помещает также ряд управляющих файлов, которые позволяют отсекать повторяющиеся журнальные сообщения. Механизм настройки параметров ядра подробнее описан в подразделе 28.4. Специальными управляющими файлами являются `/proc/sys/kernel/printk_ratelimit`, в котором указан минимальный интервал времени в секундах между записываемыми сообщениями ядра после активизирования "заглушки" (по умолчанию это значение равно 5), и `/proc/sys/kernel/printk_ratelimit_burst`, в котором указано количество групповых сообщений, вызывающее активирование заглушки (значение по умолчанию — 10). Эти параметры носят лишь рекомендательный характер и поэтому не обеспечивают абсолютной гарантии полного перекрытия потока излишних сообщений.

К сожалению, журнальная регистрация на уровне сценариев запуска системы контролируется не так хорошо, как на уровне ядра. В Red Hat Enterprise Linux имеется команда `initlog`, которая перехватывает сообщения, выдаваемые командами сценариев, и направляет их в систему Syslog. К сожалению, эту команду приходится явно указывать в сценариях, что повышает их сложность. В конечном итоге журнальные сообщения попадают в файл `/var/log/boot.log`.

В других наших дистрибутивах вообще не делаются попытки перехватывать сообщения сценариев запуска. Часть информации регистрируется отдельными командами и демонами, но основная ее часть остается "незамеченной".

Первоначально в Fedora использовалась та же система перехвата сообщений `initlog`, что и в Red Hat, но в файлах начального запуска команды передачи журнальных записей были преобразованы в комментарии. К счастью, существует центральная библиотека служебных функций, `/etc/init.d/functions`, в которой строки `initlog` можно снова активизировать.

10.3. Утилита `logrotate`: управление журнальными файлами

Прекрасная утилита `logrotate` реализует различные схемы управления журнальными файлами и является частью всех рассматриваемых нами дистрибутивов.

Конфигурационный файл утилиты `logrotate` состоит из набора спецификаций для групп журнальных файлов, которыми следует управлять. Параметры, определенные вне какой-либо спецификации (например, `errors`, `rotate` и `weekly` в показанном ниже примере), являются глобальными. Их можно переопределять для конкретного журнального файла, а также указывать несколько раз в различных частях файла, меняя стандартные установки.

Ниже представлен несколько искусственный пример такого файла.

```
# Глобальные параметры
errors errors@book.admin.com
rotate 5
weekly

# Определения и параметры ротации журнальных файлов
/var/log/messages {
   postrotate
      /bin/kill -HUP `cat /var/run/syslogd.pid`
   endscript
}

/var/log/samba/*.log {
   notifempty
   copytruncate
   sharedscripts
   postrotate
      /bin/kill -HUP `cat /var/lock/samba/*.pid`
   endscript
}
```

В этой конфигурации раз в неделю осуществляется ротация файла `/var/log/messages`. Сохраняются пять последовательных версий файла, а демон `syslogd` уведомляется о каждом шаге ротации. Журнальные файлы системы Samba (их может быть несколько) также подвергаются ротации каждую неделю, но вместо того чтобы последовательно перемещать файлы, а затем создавать новый начальный файл, сценарий последовательно копирует журнальные файлы, усекая первый файл. Сигнал `HUP` посылается демонам Samba только после того, как произойдет ротация всех журнальных файлов.

В табл. 10.2 перечислены наиболее полезные параметры файла `logrotate.conf`.

Таблица 10.2. Параметры файла `logrotate.conf`

Опция	Назначение
`compress`	Сжимать все версии журнального файла, кроме текущей
`daily`, `weekly`, `monthly`	Осуществлять ротацию журнальных файлов ежедневно, еженедельно или ежемесячно
`delaycompress`	Сжимать все версии журнального файла, кроме текущей и предыдущей
`endscript`	Этот параметр помечает конец блоков `prerotate` и `postrotate`
`errors` *почтовый_адрес*	Направлять сообщения об ошибках по указанному адресу
`missingok`	Не выдавать предупреждений об отсутствии журнального файла
`notifempty`	Не осуществлять ротацию журнального файла, если он пуст
`olddir` *каталог*	Помещать старые версии журнального файла в указанный каталог
`postrotate`	Этот параметр помечает начало блока команд, выполняемых после ротации журнального файла
`prerotate`	Этот параметр помечает начало блока команд, выполняемых перед ротацией журнального файла
`rotate` *n*	Включать в цикл ротации указанное число версий журнального файла
`sharedscripts`	Запускать сценарии один раз для всей группы журнальных файлов
`size=`*порог*	Выполнять ротацию, если размер журнального файла превышает пороговую величину (например, 100 Кбайт, 4 Мбайт)

Утилита `logrotate` обычно запускается демоном `cron` раз в день. Ее стандартный конфигурационный файл называется `/etc/logrotate.conf`, но в командной строке можно указывать сразу несколько файлов (или каталогов, содержащих конфигурационные файлы). Эта особенность широко используется в рассматриваемых нами примерах дистрибутивов, в которых каталог `/etc/logrotate.d` определен как стандартное место хранения конфигурационных файлов утилиты `logrotate`. Программные пакеты, поддерживающие эту утилиту (а таковых немало), при инсталляции выбирают схему журнальной регистрации, что существенно упрощает их администрирование.

Кроме утилиты `logrotate` дистрибутивы Debian и Ubuntu включают также более простую программу `savelog`, которая управляет ротацией отдельных файлов. Она проще утилиты `logrotate` и не использует конфигурационный файл (да и не нуждается в нем). В некоторых пакетах предпочтение отдано собственным конфигурациям, создаваемым с помощью утилиты `savelog`, а не определенным утилитой `logrotate`.

10.4. Syslog: система регистрации событий

Syslog — это полнофункциональная система регистрации событий, написанная Эриком Оллманом (Eric Allman). Она выполняет две важные функции: освобождает программистов от утомительной механической работы по ведению журнальных файлов и передает управление журнальной регистрацией в руки администраторов. До появления системы Syslog каждая программа сама выбирала схему регистрации событий, а у системных администраторов не было возможности контролировать, какая информация и где именно сохраняется.

Система Syslog отличается высокой гибкостью. Она позволяет сортировать сообщения по источникам и уровню важности и направлять их в различные пункты назначения: в журнальные файлы, на терминалы пользователей и даже на другие компьютеры. Одной из самых ценных особенностей этой системы является ее способность централизовать процедуру регистрации событий в сети.

Альтернативы системе Syslog

Хотя система Syslog — проверенное средство регистрации событий в системах UNIX и Linux, в попытках устранения ряда ее недостатков были разработаны несколько альтернативных систем. Одна из них — Syslog-ng ("Syslog, next generation" — Syslog следующего поколения) теперь по умолчанию используется в системах SUSE. С точки зрения конфигурирования она достаточно сильно отличается от Syslog, и мы не станем подробно ее описывать в этой книге. Если захотите ее использовать в системе, отличной от SUSE, она доступна по адресу `www.balabit.com`.

Система Syslog предоставляет дополнительные возможности конфигурирования, фильтрования в зависимости от содержимого сообщений, обеспечения целостности сообщений и улучшенную поддержку ограничений брандмауэра при пересылке сообщений по сети.

Система SDSC Secure Syslog (разработанная в центре суперкомпьютеров в Сан-Диего) известна также как высокопроизводительная система Syslog. Реализуя спецификации RFC3195, она предоставляет "признаваемую судебными органами" систему аудита событий. Ее разработка выполнялась специально для организаций с высоким сетевым трафиком, и система содержит множество компонентов оптимизации производительности. Ее исходный код можно загрузить из сайта `sourceforge.net/projects/sdscsyslog`.

Архитектура системы Syslog

Система Syslog состоит из трех компонентов:

- `syslogd` —демон журнальной регистрации (его конфигурационный файл называется `/etc/syslog.conf`);
- `openlog()`, `syslog()`, `closelog()` — библиотечные функции, которые передают сообщения демону `syslogd`;
- `logger` —команда пользовательского уровня, которая принимает регистрационные сообщения от интерпретатора команд.

Сначала мы рассмотрим конфигурирование демона `syslogd`, а затем поговорим о том, как взаимодействовать с системой Syslog в сценариях на языке Perl.

Демон `syslogd` запускается на этапе начальной загрузки системы и работает непрерывно. Его работой нельзя управлять посредством демона `inetd`. Программы, взаимодействующие с системой Syslog, записывают журнальные сообщения (путем вызова библиотечной функции `syslog()`) в специальный файл `/dev/log`, который представляет собой UNIX-сокет. Демон `syslogd` читает сообщения из этого файла, сверяется со своим конфигурационным файлом и пересылает сообщения адресатам.

По сигналу "отбой" (`HUP`, сигнал 1) демон `syslogd` закрывает свои журнальные файлы, перечитывает конфигурационный файл и вновь переходит в режим непрерывной

регистрации сообщений. Если файл syslog.conf изменился, нужно послать демону syslogd сигнал HUP, иначе изменения не вступят в силу. Сигнал TERM приводит к завершению работы демона.

Демон syslogd записывает свой идентификатор процесса (PID) в файл /var/run/syslogd.pid. Это облегчает передачу демону сигналов из сценариев. Например, следующая команда посылает демону сигнал отбоя:

```
# kill -HUP `/bin/cat /var/run/syslogd.pid`
```

Попытки сжать или выполнить ротацию журнального файла, который был открыт демоном syslogd для записи, приводят к непредсказуемым последствиям. Правильнее переместить старый файл, создать на его месте новый файл с такими же идентификаторами владельца и группы и правами доступа, а затем послать демону сигнал HUP. Эту процедуру легко реализовать с помощью утилиты logrotate; соответствующий пример был приведен в разделе 10.3.

Конфигурирование демона syslogd

Поведение демона syslogd зависит от содержимого файла /etc/syslog.conf. Это текстовый файл относительно простого формата. Пустые строки, а также строки, начинающиеся с символа решетки (#), игнорируются. Базовый синтаксис записей файла таков:

селектор <Tab> *действие*

Например, строка

```
mail.info   /var/log/maillog
```

обеспечивает сохранение сообщений, поступающих от почтовой системы, в файле /var/log/maillog.

Селектор обозначает программу (*средство* в терминологии системы Syslog), которая посылает журнальное сообщение, и уровень важности этого сообщения. Формат селектора выглядит следующим образом:

средство.уровень

Названия средств и уровней важности следует выбирать из стандартного списка значений; программы не могут самостоятельно составлять такие списки. Есть средства, определенные для ядра, для основных групп утилит и для локальных программ. Все остальное проходит под общим названием user (т.е. пользователь).

Селекторы могут содержать ключевые слова * и none, которые обозначают соответственно “все” и “ничего”. В селекторе разрешается через запятые перечислять группу средств. Допускается также разделение самих селекторов с помощью точки с запятой.

В общем случае селекторы объединяются методом логического сложения, т.е. указанное действие будет выполняться для сообщения, соответствующего любому из селекторов. Селектор уровня none означает исключение перечисленных в нем средств независимо от того, что указано в остальных селекторах этой же строки.

Приведем несколько примеров селекторов:

```
средство.уровень                              действие
средство1,средство2.уровень                   действие
средство1.уровень1;средство2.уровень2         действие
*.уровень                                     действие
*.уровень;плохое_средство.none                действие
```

В табл. 10.3 перечислены допустимые названия средств. В настоящее время определено 21 средство.

Таблица 10.3. Названия средств Syslog

Средство	Программы или сообщения, которые его используют
*	Все средства, кроме `mark`
`auth`	Команды, связанные с безопасностью и авторизацией
`authpriv`	Конфиденциальные авторизационные сообщения[a]
`cron`	Демон `cron`
`daemon`	Системные демоны
`ftp`	Демон `ftpd`
`kern`	Ядро
`local0-7`	Восемь разновидностей локальных сообщений
`lpr`	Система спулинга построчной печати
`mail`	Система `sendmail` и другие почтовые программы
`mark`	Метки времени, генерируемые через одинаковые промежутки
`news`	Система телеконференций Usenet
`syslog`	Внутренние сообщения демона `syslogd`
`user`	Пользовательские процессы (это установка по умолчанию, если не указано иное)
`uucp`	Устаревшее средство, которое в последних версиях системы игнорируется

[a] В действительности *все* авторизационные сообщения являются конфиденциальными. Сообщения типа `authpriv` и `auth` не должны быть открытыми для всеобщего доступа.

Демон `syslogd` сам генерирует сообщения с метками времени, которые регистрируются, если в файле `syslog.conf` для них указан селектор со средством `mark`. Метки времени помогают точно определить, когда произошел сбой, к примеру "между 3:00 и 3:20 утра", а не просто "сегодня ночью". Это необходимо при устранении проблем, возникающих регулярно (например, известны случаи, когда подобным образом удавалось выяснить, что причиной необъяснимых сбоев системы были ночные уборщицы, которые включали пылесосы в компьютерные розетки, вследствие чего срабатывала автоматическая защита сети электропитания).

Если система сильно загружена, достаточно много временных меток содержится в других журнальных сообщениях. Но так бывает не всегда, особенно в утренние часы.

В табл. 10.4 перечислены уровни важности системы Syslog (в порядке убывания важности).

Уровень сообщения определяет его важность. Границы между различными уровнями несколько размыты. Существует четкое различие между событиями, заслуживающими внимания и предупреждениями (а также между предупреждениями и сообщениями об ошибках), но трудно однозначно разграничить значения уровней `alert` и `crit`. В табл. 10.7 перечислены уровни, используемые рядом часто применяемых программ.

Таблица 10.4. Уровни важности сообщений Syslog

Уровень	Приблизительное значение
emerg	Экстренные сообщения
alert	Срочные сообщения
crit	Критические состояния
err	Другие ошибочные состояния
warning	Предупреждения
notice	Необычные события, которые заслуживают внимания
info	Информационные сообщения
debug	Отладочные сообщения

В файле `syslog.conf` уровни обозначают *минимальную* важность, которую сообщение должно иметь для того, чтобы быть зарегистрированным. Например, сообщение почтовой системы, имеющее уровень важности `warning`, будет соответствовать селектору `mail.warning`, а также селекторам `mail.info`, `mail.notice`, `mail.debug`, `*.warning`, `*.notice`, `*.info` и `*.debug`. Если в файле `syslog.conf` указано, что сообщения `mail.info` должны регистрироваться в файле, то сообщения `mail.warning` тоже будут направляться в этот файл.

В Linux-версиях системы Syslog перед названиями уровней могут ставиться символы = и !, означающие "только этот уровень" и "за исключением этого и более высоких уровней" (табл. 10.5).

Таблица 10.5. Примеры использования квалификаторов уровня в файле `syslog.conf`

Селектор	Назначение
`mail.info`	Выбор почтовых сообщений уровня `info` и выше
`mail.=info`	Выбор почтовых сообщений только уровня `info`
`mail.info;mail.!err`	Выбор почтовых сообщений уровней `info`, `notice` и `warning`
`mail.debug;mail.!=warning`	Выбор почтовых сообщений любого уровня, кроме `warning`

Поле *действие* определяет способ обработки сообщения. Возможные варианты перечислены в табл. 10.6.

Таблица 10.6. Действия системы Syslog

Действие	Назначение
имя_файла	Дописать сообщение в указанный файл на локальном компьютере
@*имя_хоста*	Переслать сообщение демону `syslogd`, выполняющемуся на компьютере с указанным именем
@*IP_адрес*	Переслать сообщение демону `syslogd` на компьютере с указанным IP-адресом
|*файл_FIFO*	Записать сообщение в указанный именованный канал[a]
пользователь1, *пользователь2*,...	Вывести сообщение на экраны терминалов указанных пользователей, если они зарегистрированы в системе
*	Вывести сообщение на экраны всех зарегистрированных пользователей

[a] Для получения дополнительной информации введите команду `info mkfifo`.

Если в качестве действия задано имя файла (или имя файла FIFO), то это должно быть полное имя. При отсутствии указанного файла демон `syslogd` создает его в процессе записи первого сообщения[2]. Перед именем файла можно ставить дефис, чтобы после записи каждого сообщения не осуществлялся системный вызов `sync`. Синхронизация помогает сохранить как можно больше журнальной информации в случае сбоя системы, но достаточно накладна с точки зрения производительности системы.

Если указывается не IP-адрес, а имя хоста, оно должно быть известно одной из служб преобразования имен, например DNS или NIS.

Хотя в селекторе можно указывать сразу несколько средств и уровней, выполнение нескольких действий не предусмотрено. Для того чтобы послать сообщение в два разных места (например, в локальный файл и на центральный регистрационный хост), нужно включить в конфигурационный файл две строки с одинаковыми селекторами.

Поскольку сообщения системы Syslog могут использоваться для организации атак типа "отказ от обслуживания", демон `syslogd` откажется принимать сообщения от других компьютеров, если только он не был запущен с флагом `-r`. По умолчанию демон также отказывается выступать в роли внешнего диспетчера сообщений: сообщения, поступающие от сетевого хоста, не могут быть пересланы на другой хост. Флаг `-h` отменяет такое поведение. Если нужно, чтобы эти опции были всегда включены, модифицируйте соответствующим образом сценарий `/etc/sysconfig/syslog` в RHEL и Fedora или `/etc/init.d/sysklogd` в Debian и Ubuntu.

Разработка схемы журнальной регистрации для конкретной организации

Для небольшой организации достаточно так сконфигурировать систему журнальной регистрации, чтобы сообщения о важных системных ошибках и предупреждения хранились в отдельном файле на каждом компьютере. В основном так было до появления системы Syslog. Файл `syslog.conf` можно адаптировать для каждого конкретного компьютера.

В большой сети необходима централизованная регистрация сообщений. Такой подход позволяет контролировать информационные потоки и скрывать журнальные данные от лица, взломавшего защиту какого-либо компьютера в сети. Хакеры часто изменяют системные журнальные файлы, чтобы замести следы. Если же журнальную информацию сразу убрать от посторонних глаз, то подделать ее будет гораздо труднее. Имеющийся в сети брандмауэр должен запрещать внешним хостам передавать сообщения демону `syslogd`.

Помните, однако, что каждый может вызвать функцию `syslog()` и создать журнальные записи, якобы посланные каким-то демоном или утилитой. Кроме того, система Syslog пользуется протоколом UDP, который не является надежным, так как не гарантирует доставку сообщений.

📖 Информацию о распространении файлов по сети можно найти в главе 17.

В качестве сервера журнальной регистрации выбирайте стабильно работающий компьютер, желательно надежно защищенный и с небольшим количеством зарегистрированных пользователей. Другие компьютеры сети могут пользоваться базовым конфигурационным файлом системы Syslog, который ведется на центральном хосте. Таким

[2] В этом плане демон ведет себя противоположно первоначальной реализации системы Syslog, в которой требовалось, чтобы журнальные файлы создавались заранее.

образом, достаточно иметь лишь две версии файла `syslog.conf`. Это обеспечивает полноту журнальной регистрации, не превращая ее в кошмар для администратора.

Для обеспечения максимальной безопасности сервер системы Syslog должен быть защищен от остальных хостов сети брандмауэром, который допускает входящие подключения только к порту Syslog, причем только от тех хостов, которым разрешена регистрация в нем. В зависимости от принятого в данной организации уровня безопасности брандмауэр может разрешать подключения со стороны рабочих станций администраторов, дабы упростить им задачу просмотра журнальных файлов.

В некоторых очень больших организациях считают необходимым ввести в иерархию журнальной регистрации дополнительные уровни. К сожалению, текущая версия системы Syslog сохраняет имя хоста-отправителя только на время одной пересылки сообщения. Если клиентский компьютер отправляет журнальные данные серверу, который, в свою очередь, пересылает их главному регистрационному хосту, то последний будет рассматривать эти сообщения как поступившие от сервера, а не от клиента.

Примеры конфигурационных файлов

Конфигурация системы Syslog — это та область, в которой дистрибутивы Linux сильно различаются. Поскольку понять содержимое файла `syslog.conf` несложно, мы не станем подробно анализировать конфигурационные файлы наших демонстрационных дистрибутивов. Гораздо полезнее будет рассмотреть некоторые распространенные схемы журнальной регистрации.

Ниже приведены примеры трех файлов `syslog.conf`, которые соответствуют автономному компьютеру, используемому в небольшой сети, клиентскому компьютеру в крупной сети и центральному регистрационному хосту этой же сети. Последний называется `netloghost`[3].

Автономный компьютер

Ниже показана базовая конфигурация для автономного компьютера.

```
# Файл syslog.conf для небольшой сети или автономной системы.
# Экстренные сообщения направляются всем зарегистрированным пользователям.
*.emerg                                 *
# важные сообщения
*.warning;daemon,auth.info;user.none    /var/log/messages
# ошибки принтера
lpr.debug                               /var/log/lpd-errs
```

В первой не являющейся комментарием строке организуется вывод экстренных сообщений на экраны терминалов всех пользователей, зарегистрированных в данный момент в системе. В качестве примера можно привести сообщения, выдаваемые командой `shutdown` перед выключением системы.

Вторая значащая строка обеспечивает запись важных сообщений в файл `/var/log/messages`. Уровень `info` ниже уровня `warning`, поэтому выражение `daemon,auth.info` включает дополнительную регистрацию сообщений, поступающих от команд `passwd`

[3] Точнее, `netloghost` — один из псевдонимов хоста. Наличие псевдонима позволяет заменять регистрационный хост с минимальными затратами на переконфигурирование системы. Псевдоним можно добавить в файл `/etc/hosts` или задать с помощью записи CNAME в базе данных DNS. Подробнее о записях CNAME рассказывается в разделе 15.7.

и su, а также от всех демонов. Третья строка предназначена для записи сообщений об ошибках принтера в файл /var/log/lpd-errs.

Сетевой клиент

Клиентский компьютер обычно пересылает важные сообщения на центральный регистрационный хост.

```
# Файл syslog.conf для клиентских компьютеров
# Экстренные сообщения направляются всем зарегистрированным пользователям.
*.emerg;user.none                       *

# Важные сообщения пересылаются на центральный компьютер.
*.warning;lpr,local1.none               @netloghost
daemon,auth.info                        @netloghost

# Сообщения от локальных программ также отправляются на центральный компьютер.
local2.info;local7.debug                @netloghost

# Сообщения об ошибках принтера хранятся локально.
lpr.debug                               /var/log/lpd-errs

# Сообщения утилиты sudo регистрируются от имени средства local2.
# Сохраняем их копии.
local2.info                             /var/log/sudo.log

# Сообщения ядра также хранятся локально.
kern.info                               /var/log/kern.log
```

В такой конфигурации локально хранится не очень много информации. Если компьютер netloghost выключен или недоступен, регистрационные сообщения будут безвозвратно теряться. На этот случай можно создавать локальные копии важных сообщений.

В системе, где инсталлирован большой объем локального программного обеспечения, неоправданно много сообщений будет регистрироваться от имени средства user на уровне emerg. В показанном примере такая ситуация заведомо исключается благодаря выражению user.none в первой значащей строке.

📖 Подробнее об утилите sudo рассказывалось в разделе 3.4.

Вторая и третья значащие строки служат для пересылки всех важных сообщений на центральный регистрационный сервер; исключение составляют сообщения подсистемы печати и университетской (локальной) системы доступа по магнитным карточкам. В четвертой строке организуется отправка локальной журнальной информации также на центральный компьютер. Последние три строки обеспечивают сохранение локальных копий сообщений об ошибках принтера, журнальных сообщений утилиты sudo и сообщений ядра.

Центральный регистрационный хост

Ниже показана конфигурация компьютера netloghost — центрального защищенного регистрационного хоста сети среднего размера, включающей около 7000 компьютеров.

```
# Файл syslog.conf для главного регистрационного хоста.
# Экстренные сообщения направляются на консоль и в журнальный файл,
# дополняемые метками времени.
*.emerg                                    /dev/console
*.err;kern,mark.debug;auth.notice          /dev/console
*.err;kern,mark.debug;user.none            /var/log/console.log
auth.notice                                /var/log/console.log

# Сообщения, не являющиеся экстренными, записываются в обычные журнальные файлы.
*.err;user.none;kern.debug                 /var/log/messages
daemon,auth.notice;mail.crit               /var/log/messages
lpr.debug                                  /var/log/lpd-errs
mail.debug                                 /var/log/mail.log

# В следующих строках обрабатываются сообщения локальных программ
# авторизации, таких как sudo и npasswd.
local2.debug                               /var/log/sudo.log
local2.alert                               /var/log/sudo-errs.log
auth.info                                  /var/log/auth.log

# другие локальные программы
local4.notice                              /var/log/da.log
local7.debug                               /var/log/tcp.log

# пользовательские процессы (стандартное правило обработки сообщений,
# для которых не задано средство)
user.info                                  /var/log/user.log
```

Сообщения, поступающие от локальных программ и демонов `syslogd` по сети, записываются в журнальные файлы. В некоторых случаях выходная информация каждого средства помещается в отдельный файл.

Центральный регистрационный компьютер генерирует метки времени для каждого получаемого сообщения. Это время не всегда совпадает со временем отправки сообщения. Если в систему входят компьютеры, расположенные в разных часовых поясах, или системные часы не синхронизированы, информация о времени окажется недостоверной.

Пример выходной информации системы Syslog

Ниже показан фрагмент одного из журнальных файлов, расположенных на главном регистрационном хосте факультета вычислительной техники университета штата Колорадо.

```
Dec 18 15:12:42 av18.cs.colorado.edu sbatchd[495]: sbatchd/main: ls_info()
       failed: LIM is down; try later; trying …
Dec 18 15:14:28 proxy-1.cs.colorado.edu pop-proxy[27283]: Connection from
       128.138.198.84
Dec 18 15:14:30 mroe.cs.colorado.edu pingem[271]: maltese-
       office.cs.colorado.edu has not answered 42 times
Dec 18 15:15:05 schwarz.cs.colorado.edu vmunix: Multiple softerrors: Seen 100
       Corrected Softerrors from SIMM J0201
Dec 18 15:15:16 coyote.cs.colorado.edu PAM_unix[17405]: (sshd) session closed
       for user trent
Dec 18 15:15:48 proxy-1.cs.colorado.edu pop-proxy[27285]: Connection from
       12.2.209.183
Dec 18 15:15:50 av18.cs.colorado.edu last message repeated 100 times
```

В этом примере представлена информация, поступившая от нескольких компьютеров (`av18`, `proxy-1`, `schwarz`, `mroe` и `coyote`) и от нескольких программ: `sbatchd`, `pop-proxy`, `pingem` и библиотеки PAM.

Обратите внимание на последнюю строку, в которой говорится о сообщении, повторившемся 100 раз. Чтобы сократить объем журнальных файлов, система Syslog обычно пытается объединять дублирующиеся сообщения, заменяя их резюмирующей строкой. Однако компьютер, с которого был взят этот файл, принимает журнальные сообщения от множества других компьютеров, поэтому такое сообщение может сбить читателей с толку. Поясним: оно ссылается на предыдущее сообщение от компьютера `av18`, а не на предыдущее сообщение в журнальном файле.

Рекомендуется регулярно просматривать журнальные файлы. Определите, какое их состояние является нормальным, чтобы в случае отклонения от нормы сразу же заметить это. А еще лучше инсталлировать программу обработки журнальных файлов, например `swatch`, которая будет делать то же самое автоматически (об этом рассказывается в разделе 10.5).

Программы, использующие систему Syslog

В табл. 10.7 кратко описаны программы, работающие с системой Syslog, а также перечислены соответствующие им средства и уровни.

Таблица 10.7. Программы, использующие систему Syslog

Программа	Средство	Уровни	Описание
`cron`	Демон `cron`	`info`	Демон планирования заданий
`cups`	`lpr`	`info-err`	Общая система вывода на печать UNIX
`ftpd`	`ftp`	`debug-crit`	Демон FTP (`wu-ftpd`)
`inetd`	`daemon`	`warning`, `err`	Супердемон Internet (Debian)
`imapd`	`mail`	`info-alert`	Почтовый сервер IMAP
`login`	`authpriv`	`info-err`	Программа входа в систему
`lpd`	`lpr`	`info-err`	Система вывода на печать BSD
`named`	`daemon`	`info-err`	Сервер имен (DNS)
`ntpd`	`daemon, user`	`info-crit`	Демон сетевого времени
`passwd`	`auth`	`notice`, `warning`	Программа установки паролей
`popper`	`local0`	`debug`, `notice`	Почтовый сервер POP3
`sendmail`	`mail`	`debug-alert`	Система доставки почты
`ssh`	`auth`	`info`	Защищенный интерпретатор команд (для удаленного входа в систему)
`su`	`auth`	`notice`, `crit`	Утилита переключения идентификаторов пользователей
`sudo`	`local2`	`notice`, `alert`	Ограниченная версия утилиты `su`
`syslogd`	`syslog, mark`	`info-err`	Демон обработки внутренних ошибок, меток времени и др.
`tcpd`	`local7`	`debug-err`	TCP-оболочка для демона `inetd`
`vmlinuz`	`kern`	*все*	Ядро
`xinetd`	*конфигурируется*	`info` (по умолчанию)	Разновидность демона `inetd` (Red Hat, SUSE)

Располагая приведенной информацией, можно, казалось бы, безошибочно определить, какие сообщения следует сохранять, а какие — игнорировать. Но это не совсем так. На практике нужно тщательно изучить, какие уровни регистрации важны для системы. Лучше начать с регистрации всех сообщений и постепенно отсеивать ненужные, пока объем передаваемых данных не станет приемлемым.

Отладка системы Syslog

Команда `logger` позволяет выдавать журнальные сообщения в сценариях интерпретатора команд. Кроме того, эту команду можно использовать для проверки изменений в конфигурационном файле демона `syslogd`. Например, если в файл была добавлена строка

```
local5.warning     /tmp/evi.log
```

и нужно убедиться в том, что она действительна, введите команду

```
$ logger -p local5.warning "test message"
```

Строка, содержащая фразу “test message” (тестовое сообщение), должна быть записана в файл `/tmp/evi.log`. Если этого не произошло, то, возможно, демону `syslogd` не был послан сигнал отбоя.

Очень осторожно подходите к выводу журнальной информации на системную консоль, `/dev/console`, а также на любой псевдотерминал или порт, где поддерживается управление потоками данных. Если кто-то случайно нажмет на консоли клавиши <Ctrl+S>, вывод на консоль прекратится. В результате все вызовы функции `syslog()` окажутся заблокированными, и система начнет работать очень медленно. Хороший способ проверки подобной ситуации — послать сообщение на консоль с помощью команды `logger`. Если команда зависает, нужно найти проблемный порт, нажать <Ctrl+Q> и пересмотреть схему журнальной регистрации.

Вывод журнальных сообщений на консоль опасен еще и тем, что при возникновении серьезной проблемы поток сообщений “парализует” консоль именно тогда, когда она больше всего нужна. При использовании некоторых типов кадровых буферов и неоптимизированных драйверов консоли поток сообщений может полностью парализовать всю систему.

Кроме того, в некоторых конфигурациях (например, если управление консолью осуществляется через консольный сервер) появляется риск атаки со стороны хакеров.

Использование функций системы Syslog в сценариях

Библиотечные функции `openlog`, `syslog` и `closelog` позволяют программам взаимодействовать с системой Syslog. Существуют версии этих функций для языков C, Perl, Python, и PHP. Мы же рассмотрим лишь программирование Perl-сценариев.

Чтобы импортировать определения библиотечных функций, нужно включить в начало сценария следующую строку:

```
use Sys::Syslog;
```

Функция `openlog` инициализирует систему регистрации сообщений, передавая ей заданное имя средства:

```
openlog(идентификатор, опции_регистрации, средство);
```

Сообщения регистрируются с указанными опциями и начинаются с идентификационной строки, переданной в первом аргументе. Если функция `openlog` не используется, то по умолчанию идентификационная строка равна текущему имени пользователя, в качестве опций задается пустая строка, а названием средства становится строка "user". Список возможных опций приведен в табл. 10.8.

Таблица 10.8. Опции регистрации сообщений для функции `openlog`

Опция	Назначение
`pid`	Включать идентификатор текущего процесса в каждое регистрируемое сообщение
`ndelay`	Подключаться к демону `syslogd` немедленно (не ожидая доставки сообщения)
`cons`	Посылать сообщения на системную консоль, если демон `syslogd` недоступен
`nowait`	Не вызывать функцию `wait(3)` для дочерних процессов, созданных с помощью функции `fork` с целью записи сообщений на консоль

К примеру, вызов функции `openlog` может выглядеть так:

```
openlog("adminscript", "pid,cons", "local4");
```

Библиотечная функция `syslog` посылает сообщение демону `syslogd`, который регистрирует его с заданным приоритетом:

```
syslog(приоритет, сообщение, ...);
```

В журнальном файле к сообщению добавляются дата и время его получения, а также имя компьютера и строка идентификации, заданная в функции `openlog`. После аргумента сообщение могут следовать другие параметры. Формат сообщения напоминает спецификацию функции `printf`, включающую текст и значения других переменных, например:

```
syslog("info", "Delivery to '%s' failed after %d attempts.", $user, $nAttempts);
```

Особый спецификатор `%m` заменяется сообщением об ошибке, номер которой извлекается из переменной `errno` (в ней хранится код последней ошибки).

Строка приоритета вида "*уровень*|*средство*" задает уровень важности и название средства. Если функция `openlog` не была вызвана и строка идентификации не задана явно, то функция `syslog` проверяет, выглядит ли второй аргумент как стандартное сообщение об ошибке, например:

```
adminscript: User "nobody" not found in /etc/passwd file.
```

Если проверка прошла успешно, то часть строки перед двоеточием принимается в качестве идентификационной строки. Эта удобная (хотя и недокументированная) особенность делает вызов функции `openlog` ненужным. Тем не менее, не следует от этого отказываться, так как лучше задать название средства один раз (в функции `openlog`), чем постоянно повторять его в тексте программы.

Функция `closelog` закрывает канал регистрации сообщений:

```
closelog();
```

Ее нужно вызывать, если требуется повторно инициализировать систему журнальной регистрации с другими опциями. Хорошим стилем считается также вызов функции `closelog` при завершении работы программы.

Приведем полный пример сценария:

```
use Sys::Syslog;

openlog("adminscript", "cons,pid", "user");
syslog("warning", "Those whom the gods would destroy, they first teach Basic.");
closelog();
```

Сценарий выдаст сообщение следующего вида (191 — идентификатор процесса, соответствующий сценарию adminscript):

```
Dec 28 22:56:24 moet.colorado.edu adminscript[191]: Those whom the gods
       would destroy, they first teach Basic.
```

10.5. Поиск полезной информации в журнальных файлах

Система Syslog прекрасно подходит для сортировки и маршрутизации журнальных сообщений, но в результате все равно получается большой набор журнальных файлов. Они могут содержать много важной информации, но ее бывает не так-то просто найти. Необходимо дополнительное программное обеспечение для анализа журнальных файлов и поиска в них нужных сообщений.

В этом сегменте рынка предлагается много бесплатных продуктов, и большинство из них работает примерно одинаково: программа сканирует недавние записи журнального файла, сравнивает их с регулярными выражениями из базы данных и обрабатывает важные сообщения. Одни программы направляют пользователю отчет по электронной почте, другие можно настраивать на нужные действия. Различия между программами проявляются в степени гибкости и размере готовых баз данных с шаблонами.

Двумя наиболее распространенными программами обработки журнальных файлов являются утилиты swatch и logcheck. Их можно загрузить из сайта sourceforge.net (программа logcheck распространяется в составе пакета sentrytools: sourceforge.net/projects/sentrytools).

Утилита swatch представляет собой Perl-сценарий, работающий в соответствии с директивами конфигурационного файла. Синтаксис этого файла достаточно гибок, так как позволяет выполнять все поддерживаемые в языке Perl операции сравнения с шаблонами. Утилита swatch может обработать конфигурационный файл за один вызов, но обычно она выполняется в фоновом режиме, отслеживая новые сообщения по мере их поступления, подобно команде tail -f. Недостатком является то, что конфигурационный файл, по сути, нужно создавать с нуля. Утилита не “знает” об особенностях работы системы и о том, какие журнальные сообщения в ней генерируются.

Утилита logcheck — более простой сценарий интерпретатора sh. В дистрибутив входит также программа на языке C, которая помогает утилите запоминать текущую позицию в журнальном файле. Благодаря этому у сообщения меньше шансов пройти незамеченным на этапе загрузки или останова системы. Утилита может запускаться периодически с помощью демона cron, а не выполняться постоянно.

С утилитой logcheck поставляются базы шаблонов для нескольких версий UNIX и Linux. Даже если в самой утилите нет особой надобности, стоит просмотреть эти базы данных, так как в них могут содержаться очень интересные и полезные шаблоны.

Недостаток обеих утилит заключается в том, что за раз они обрабатывают только один журнальный файл. Если система Syslog направляет сообщения одновременно в

несколько файлов, приходится дублировать некоторые из сообщений в каком-нибудь центральном файле, который часто обнуляется, а затем передавать его на последующую обработку одному из сценариев. Это проще, чем организовывать сложную сеть сценариев, управляющих несколькими журнальными файлами.

Стоит также упомянуть об утилите `logwatch`. Она представляет собой скорее средство записи итогового журнала, чем средство оперативного мониторинга и выдачи предупреждений, но ее преимуществом является относительная простота и доступность в виде пакетов RPM и APT. Кроме того, она по умолчанию устанавливается в системах Fedora и Red Hat.

Утилита `SEC` (Simple Event Correlator — простой коррелятор событий) представляет собой иной тип средства управления журнальными файлами. Она является сценарием Perl, который считывает строки из файлов, именованных каналов или из стандартного ввода и преобразует их в различные классы "входных событий", сравнивая их с регулярными выражениями. Затем в соответствии с правилами, определенными в конфигурации, программа преобразует входные события в такие выходные события, как выполнение конкретного сценария или передача сообщения указанному каналу или файлу.

Дистрибутив `SEC` с подробной man-страницей и множеством примеров доступен по адресу `kodu.neti.ee/~risto/sec`. Дополнительные примеры доступны на Web-сайте. Пакет `SEC` не является "полностью готовой к использованию" утилитой, как названные выше программы, но он может послужить хорошей отправной точкой для построения специализированного средства анализа журнальных данных.

Независимо от того, какая программа используется для сканирования журнальных файлов, есть ряд общих закономерностей.

- Большинство сообщений, связанных с безопасностью, должно просматриваться немедленно. Важно отслеживать неудачные попытки входа в систему, равно как и неудачные вызовы команд `su` и `sudo`, чтобы предотвратить возможный взлом системы, пока еще не поздно. Если кто-то просто неправильно ввел свой пароль (как чаще всего бывает), то появление оперативной подсказки произведет хорошее впечатление и повысит репутацию системного администратора.
- Сообщения о нехватке места на диске должны помечаться и немедленно обрабатываться. Когда диск переполнен, работать с ним невозможно.
- Многократно повторяемые сообщения заслуживают внимания, хотя бы в профилактических целях.

10.6. Упражнения

10.1. Зачем хранить старые журнальные файлы?

10.2. В чем разница между файлами `lastlog` и `wtmp`? Какой должна быть схема ротации для каждого из них?

10.3. Проанализируйте следующую запись файла `syslog.conf`:

```
*.notice;kern.debug;lpr.info;mail.crit;news.err /var/log/messages
```

Есть ли в ней смысл?

10.4. Поищите в журнальных файлах записи демона `named` (сервер имен DNS). Есть ли в сети компьютеры, пытающиеся динамически обновлять доменные файлы? Успешны ли эти попытки? (Может потребоваться доступ с правами суперпользователя.)

☆ 10.5. Где расположен журнал начальной загрузки на вашем компьютере? В чем трудности журнальной регистрации на этапе начальной загрузки? Как эти трудности решаются демоном `klogd`?

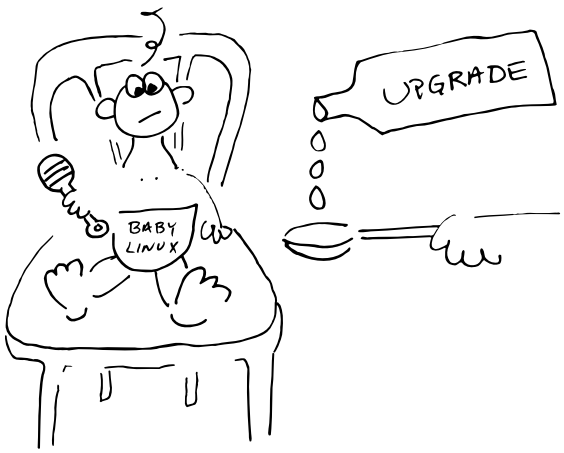
☆ 10.6. Проанализируйте схему журнальной регистрации, используемую в вашей организации, включая схему ротации журнальных файлов. Сколько дискового пространства отведено для журнальных файлов? Насколько долго хранятся журнальные файлы? Можно ли заранее предсказать ситуации, когда используемых схем окажется недостаточно? Какое решение можно порекомендовать? (Необходим доступ с правами суперпользователя.)

☆ 10.7. Некоторые журнальные сообщения являются чрезвычайно важными и должны немедленно просматриваться системным администратором. Что можно предпринять для того, чтобы это произошло как можно скорее?

☆ 10.8. Напишите программу или сценарий для выдачи журнальных сообщений от имени средства `user`. (Может потребоваться доступ с правами суперпользователя.)

Глава 11

Управление программным обеспечением и конфигурацией



Несмотря на то что дистрибутивы Linux быстро становятся более функциональными и дружественными по отношению к пользователям, управление конфигурацией программного обеспечения по-прежнему остается важным делом в администрировании. Помимо выполнения стандартных действий при инсталляции, вам придется делать настройки, подходящие для вашей организации. Администраторам обычно приходится выполнять все перечисленные ниже задачи:

- выполнять автоматизированную инсталляцию операционной системы на группу компьютеров;
- осуществлять настройку систем в локальных средах;
- следить за выходом "заплат" и своевременно обновлять с их помощью систему;
- управлять добавочными программными пакетами.

Процесс конфигурирования готового дистрибутива или программного пакета, направленный на удовлетворение всех ваших потребностей (и не допускающий нарушения локальных условий защиты, размещения файлов и топологии сети), часто называют "локализацией". В этой главе рассматриваются методики, которые позволяют упростить инсталляцию программного обеспечения, в том числе в крупных системах.

11.1. Базовая инсталляция Linux

Процедура инсталляции всех современных дистрибутивов Linux довольно проста. Необходимо загрузиться с компакт-диска или дискеты, ответить на ряд вопросов и указать программе-инсталлятору, какие пакеты нужно скопировать на жесткий диск. Многие дистрибутивы сопровождаются хорошими руководствами по инсталляции.

Версии Red Hat Enterprise сопровождаются печатной версией руководства по инсталляции. Эти и другие руководства по Red Hat можно найти по адресу `www.redhat.com/docs/manuals`.

Fedora имеет свой собственный проект документации, однако в связи с тем, что текущие версии Fedora тесно связаны с Red Hat Enterprise Linux, добровольцы от Fedora, судя по всему, не собираются тратить время на дублирование содержимого существующей документации по этой версии Red Hat. Ответы на вопросы по Fedora обычно можно найти в руководствах по Red Hat.

К коробочной версии SUSE Linux Enterprise тоже прилагаются печатные руководства. Их электронные версии можно найти в сети Internet по адресу `www.novell.com/documentation/SUSE.html`. Аналогичную документацию по openSUSE можно найти по адресу `www.opensource.org/Documentation`.

Руководство по инсталляции Debian находится по адресу `http://www.debian.org/releases/stable/installmanual`.

Материалы по инсталляции Ubuntu можно найти на страницах Internet-проекта Wiki по адресу: `https://wiki.ubuntu.com/Installation`.

Любой, кому придется инсталлировать систему сразу на несколько компьютеров, столкнется с недостатками инсталляции в интерактивном режиме. Прежде всего, выполняемые действия очень неудобно многократно повторять. (Сможете ли вы повторно выбрать непротиворечивое множество программных пакетов из той сотни, которая предлагается вам во время инсталляции?) Свести к минимуму “ошибки пилота” (так называют неверное конфигурирование или неправильное использование программы) можно благодаря контрольному списку локализации, хотя даже его использование не сможет полностью защитить от всевозможных ошибок. К тому же, инсталляция в интерактивном режиме отнимает много времени и сил.

Инсталляторы всех наших демонстрационных дистрибутивов допускают возможность написания инсталляционных сценариев, поэтому существует вариант создать дискету или компакт-диск, на которой записана требуемая конфигурация. Тем самым решается проблема многократного повторения инсталляции, но не проблема масштабируемости. Когда система инсталлируется на сотни компьютеров, придется использовать сотни конфигурационных дискет — приятного в этом, согласитесь, мало.

Автоматическую инсталляцию можно производить и по сети. Между прочим, этот вариант является самым удобным для тех организаций, в которых приходится модернизировать сотни или тысячи систем. Все наши демонстрационные дистрибутивы можно инсталлировать именно таким образом, однако готовые системы перед этим нужно будет соответствующим образом сконфигурировать.

Также, в некоторых проектах с открытым исходным кодом стараются применять сетевые инсталляторы, разработанные сторонними фирмами (в качестве примера можно привести SystemImager). Об этих системах мы будем говорить уже в этой главе.

Загрузка по сети на персональном компьютере

На персональном компьютере систему можно полностью загрузить по сети, а не с жесткого диска, дискеты или компакт-диска. Это очень удобный способ, так как администратору не нужно присутствовать при загрузке, чтобы вставлять в приемник дисковода компакт-дисков загрузочный носитель.

Производители UNIX-систем поддерживают загрузку по сети довольно давно, однако для персональных компьютеров эта особенность была стандартизирована недавно. После нескольких лет создания загрузочных ПЗУ вручную (см. `etherboot.sourceforge.net`) фирма Intel разработала стандарт PXE (Pre-boot eXecution Environment — предзагрузочная среда выполнения). Его нельзя назвать совершенным, но лучше такой стандарт, чем вообще никакого.

PXE действует подобно миниатюрной операционной системе, размещенной в схеме ПЗУ на вашей сетевой плате. Он предлагает системе BIOS использовать его сетевые особенности посредством стандартизированного API. При таком взаимодействии один загрузчик может загружать по сети систему Linux на любом персональном компьютере, понимающем стандарт PXE, не устанавливая при этом какие-либо специальные драйвера для каждой сетевой платы.

Внешняя (сетевая) часть протокола PXE не содержит никаких сложностей и подобна процедурам загрузки по сети, используемых в других архитектурах. Компьютер осуществляет широковещательную рассылку специального DHCP-запроса, а DHCP-сервер или прокси-сервер возвращает DHCP-пакет, содержащий значения PXE-параметров (имя загрузочного сервера плюс имя загрузочного файла). Клиент получает от сервера свой загрузочный файл по протоколу TFTP (возможно использование многоадресной версии этого протокола), после чего запускает его.

📖 Более подробно о DHCP можно почитать в разделе 12.9.

Спецификация PXE не обеспечивает полную обратную совместимость с протоколом DHCP, но на практике многие современные DHCP-серверы предоставляют услуги загрузки по протоколу PXE.

Степень поддержки PXE, которой обладают инсталляторы, у разных дистрибутивов разная. Однако в этой области ведутся активные разработки, и функциональные системы, построенные на основе PXE, быстро приобретают широкое распространение.

Документы, описывающие PXE, можно найти в разных местах. Для начала неплохо ознакомиться с документом *Remote Boot HOWTO*, который доступен по адресу `www.tdlp.org`.

Конфигурирование протокола PXE в Linux

Существует несколько загрузочных систем на основе PXE, однако самой безотказной в настоящее время является система PXELINUX, разработанная Питером Анвином (H. Peter Anvin). Она является частью комплекта загрузчиков SYSLINUX, разработанного им же, который может пригодиться на все случаи жизни. Домашняя страница этой системы расположена по адресу `syslinux.zytor.com`.

PXELINUX содержит загрузочный файл, который инсталлируется в каталог `tftpboot` сервера, и который загружается на загружаемый ПК во время работы PXE. ПК выполняет загрузочный файл и загружает его конфигурацию с сервера; конфигурация определяет, какое ядро необходимо использовать. Эта цепочка событий может проистекать без вашего вмешательства; как вариант, можно создать специальное меню загрузки.

PXELINUX использует для своих загрузок PXE API и, поэтому, не зависит от аппаратной части в течение всего процесса загрузки. Более того, она может загружать не только Linux, но и другие операционные системы, а также может загружать образы дискет, если вы используете ядро MEMDISK, которое тоже входит в состав пакета SYSLINUX.

На стороне сервера следует использовать DHCP-сервер организации ISC. Если применяется другой DHCP-сервер, вам будет необходимо или сконфигурировать дополнительные параметры DHCP, или использовать числовые представления в конфигурационном файле сервера.

В главе 14 книги *Red Hat Enterprise Linux 3 System Administration Guide* рассказано все о протоколе PXEP и загрузке по сети. Ее можно найти по адресу `http://www.redhat.com/docs/manuals`.

Fedora, ближайшая бесплатная "родственница" Red Hat, позволяет работать с графической системой конфигурации по сети. Для этого нужно выполнить команду `system-config-netboot`.

В SUSE нет никаких PXE-пакетов, но можно взять их из Red Hat. Программа YaST легко конфигурируется на работу по протоколу PXE, хотя официально эта возможность не поддерживается.

Инсталлятор Debian можно загружать по сети без каких-либо проблем; нужно лишь загрузить образ ядра загрузки по сети с адреса `www.debian.org/devel/debian-installer`. А еще есть страница на сайте проекта Wiki, на которой перечислены доступные ресурсы: `wiki.debian.net/index.cgi?DebianInstallerNetbootPXE`.

На момент написания этой книги официальное руководство по инсталляции Ubuntu для версии 6.06 еще не было готово. Между тем, вы можете ознакомиться с его черновиком на странице по адресу `doc.ubuntu.com`. Чтобы быстро ознакомиться с опциями, нужно войти в разделы Installation/Netboot и Installation/LocalNet на странице проекта Wiki: `https://wiki.ubuntu.com/Installation/Netboot`.

Дистанционная загрузка на специализированных компьютерах

PXE — это протокол компании Intel, и его применение ограничено платформами IA-32 и IA-64. На других платформах применяются иные методы загрузки по сети, которые почти всегда более элегантные, чем протокол PXE. Интересно, что сейчас, когда Linux часто инсталлируется на компьютеры с архитектурой, отличной от Intel, многие из "специализированных" UNIX-систем теперь имеют вариант, при котором по сети загружается Linux, а не "родная" операционная система.

Обсуждение различий между архитектурами выходит за рамки книги; укажем лишь, где найти соответствующую информацию.

Компьютеры SPARC и большинство систем PowerPC используют программу Open Firmware, которую легко загрузить по сети (достаточно ввести `boot net`). Хорошим руководством по дистанционной загрузке Linux на процессорах SPARC является ресурс UltraLinux FAQ (`www.ultralinux.org`). На компьютерах Alpha для загрузки Linux применяется механизм SRM, описанный в документе *SRM HOWTO*. Для компьютеров Macintosh, RS/6000 и других систем на базе PowerPC процедура дистанционной загрузки

зависит как от оборудования, так и от имеющегося загрузчика, поэтому ищите информацию в документации к загрузчику. На компьютерах AMD64 PXE используется точно так же, как и в подобных им компьютерах семейства x86.

Kickstart: автоматизированный инсталлятор Red Hat

Kickstart — это утилита, предназначенная для автоматической инсталляции программного обеспечения Red Hat. Она представляет собой интерфейс к стандартному инсталлятору Red Hat — программе Anaconda — и зависит как от версии дистрибутива, так и от имеющихся пакетов RPM. В отличие от SystemImager и других программ копирования низкоуровневых образов, утилита Kickstart достаточно гибкая и автоматически распознает аппаратное обеспечение системы.

Если в вашей среде необходимо внести множество локальных настроек, то эти настройки можно описать в сценарии, который запускается по окончании инсталляции. Этот сценарий настроит систему в соответствии с вашими локальными стандартами. Недостаток такого способа заключается в том, что этот сценарий быстро разрастается, становясь слишком громоздким и неудобным для администрирования. Другой способ заключается в создании пакета RPM, содержащего перечень локальных настроек. Мы рекомендуем второй вариант, так как он позволяет легко управлять версиями, упрощает последующие обновления, дает возможность контролировать взаимозависимости и обеспечивает все остальные преимущества системы управления пакетами.

О пакетах подробно рассказывается далее в этой главе.

Создание файла конфигурации

Работа утилиты Kickstart контролируется конфигурационным файлом, который обычно называется `ks.cfg`. Формат этого файла довольно прост. Для тех, кто не любит работать с текстовыми файлами, существует графическая утилита `redhat-config-kickstart` в Enterprise Linux и `system-config-kickstart` в Fedora, упрощающая настройку конфигурационного файла[1].

Файл `ks.cfg` можно также очень легко генерировать программным путем. Предположим, что на серверах и клиентах необходимо инсталлировать разный набор пакетов; предположим также, что у вас есть два офиса, и вам нужно, чтобы система была настроена в них по-разному. Тогда следует написать небольшой Perl-сценарий, который с помощью соответствующих параметров будет генерировать конфигурационные файлы для серверов и клиентов каждого офиса. В случае изменения набора пакетов достаточно будет отредактировать Perl-сценарий, а не многочисленные конфигурационные файлы. Бывают также ситуации, когда для каждого компьютера нужен свой конфигурационный файл. Тогда тем более необходимо автоматизировать создание таких файлов.

Подробное руководство по утилите Kickstart (включая список всех возможных опций) можно найти в разделе 2 руководства *The Official Red Hat Linux Customization Guide*, которое доступно по адресу `http://www.redhat.com/docs/manuals`.

[1] Обратите внимание на то, что в Fedora все инструменты `redhat-config-*` были переименованы в `system-config-*`. Изменение имени — это еще один шаг к тому, чтобы отстранить Fedora от торговой марки Red Hat; это пока что не свидетельствует о базовых технических отличиях.

Необходимо сделать одно замечание по поводу конфигурационного файла: если при его редактировании допущена ошибка, будет выдана отладочная информация интерпретатора Python, в которой не так-то легко найти истинную причину ошибки. Одним из главных преимуществ использования `redhat-config-kickstart` или `system-config-kickstart` для генерирования конфигурационного файла является то, что в полученном файле гарантированно не будет синтаксических ошибок.

Конфигурационный файл утилиты Kickstart состоит из трех упорядоченных частей. Первая из них — это секция команд, где задаются такие установки, как язык, раскладка клавиатуры и часовой пояс. Здесь же с помощью параметра `url` задается источник, из которого загружается дистрибутив (в показанном ниже примере это компьютер `installserver`):

```
text
lang en_US                    # Язык, используемый в процессе
                              # инсталляции...
langsupport en_US             # ... и на этапе выполнения.
keyboard us                   # Американская раскладка клавиатуры.
timezone --utc America/EST    # Аргумент --utc означает, что часы
                              # синхронизируются по Гринвичу
mouse
rootpw пароль
reboot                        # Перезагрузка после инсталляции.
                              # Это почти всегда необходимо.
bootloader --location=mbr     # Стандартный загрузчик инсталлируется
                              # в главную загрузочную запись.
install                       # Новая система инсталлируется,
                              # а не обновляется.
url --url http://installserver/redhat
clearpart --all –initlabel   # Удаляются все существующие разделы
part / --fstype ext3 --size 4096
part swap --size 1024
part /var --fstype ext3 -size 1 --grow
network --bootproto dhcp
auth --useshadow --enablemd5
firewall --disabled
xconfig --defaultdesktop=GNOME --startxonboot
       --resolution 1280x1024 --depth 24
```

По умолчанию утилита Kickstart работает в графическом режиме, что препятствует ее автоматическому запуску. Ключевое слово `text` в начале файла исправляет данную ситуацию.

Параметр `rootpw` задает пароль суперпользователя на новом компьютере. По умолчанию пароль указывается в текстовом виде, что создает проблему с точки зрения безопасности. Воспользуйтесь опцией `--iscrypted`, чтобы задать зашифрованный пароль. Правда, пароли MD5 не поддерживаются.

Директивы `clearpart` и `part` задают список разделов с указанием их размеров. С помощью опции `--grow` можно отвести всю оставшуюся область жесткого диска под один из разделов. Так можно облегчить подгонку систем с разными размерами жестких дисков.

Во второй секции файла приведен список инсталлируемых пакетов, начинающийся с директивы `%packages`. В списке могут быть указаны отдельные пакеты, коллекции (например, `@ GNOME`) либо запись `@ Everything`, обозначающая весь имеющийся набор

пакетов. В первом случае задается лишь имя пакета, без номера версии и расширения `.rpm`. Приведем пример:

```
%packages
@ Networked Workstation
@ X Window System
@ GNOME
mylocalpackage
```

В третьей секции конфигурационного файла указывается произвольный набор команд интерпретатора, которые подлежат выполнению утилитой Kickstart. Существует два возможных набора команд. Один из них начинается с директивы `%pre` и содержит команды, выполняемые перед началом инсталляции. Команды, выполняемые по завершении инсталляции, помечаются директивой `%post`. В обоих случаях возможности системы по распознаванию имен компьютеров ограничены, так что лучше пользоваться IP-адресами. Кроме того, команды второго набора выполняются в среде с измененным корневым каталогом, поэтому они не имеют доступа к инсталляционному носителю.

Создание сервера Kickstart

Утилита Kickstart ожидает, что ее инсталляционные файлы расположены так же, как и на инсталляционном компакт-диске. На сервере ее пакеты должны находиться в каталоге `RedHat/RPMS` или `Fedora/RPMS`. В этот каталог разрешается добавлять собственные пакеты. Но есть ряд нюансов, о которых следует помнить.

Главная особенность заключается в том, что, получив команду инсталлировать все пакеты (запись `@ Everything` в секции пакетов файла `ks.cfg`), утилита сначала устанавливает базовые пакеты, а затем — пользовательские пакеты, причем в алфавитном порядке. Если какой-то пользовательский пакет зависит от других пакетов, не входящих в базовый набор, назовите его наподобие `zzmypackage.rpm`, чтобы он инсталлировался последним.

Если не требуется инсталлировать все пакеты, укажите нужные в секции `%packages` файла `ks.cfg` либо добавьте их в одну или несколько коллекций. Коллекции обозначаются записями вида `@ GNOME` и представляют собой предопределенные наборы пакетов, компоненты которых перечислены в файле `RedHat/base/comps` или `Fedora/base/comps` на сервере. К сожалению, формат этого файла плохо документирован. Коллекции задаются в строках, которые начинаются с цифры 0 или 1; единица указывает на то, что коллекция выбрана по умолчанию. Менять стандартные коллекции нежелательно. Мы рекомендуем оставить их в том виде, в котором они определены в Red Hat, и указывать дополнительные пакеты непосредственно в файле `ks.cfg`.

Задание пользовательского конфигурационного файла

После создания конфигурационного файла необходимо заставить утилиту Kickstart использовать его. Это можно сделать несколькими способами. Официальный способ — загрузиться с дискеты или компакт-диска и запросить инсталляцию Kickstart, введя `linux ks` в строке приглашения `prompt:`. Если не указать дополнительных аргументов, система определит свой сетевой адрес по протоколу DHCP. Затем она узнает имя загрузочного DHCP-сервера и загрузочного файла, попытается смонтировать серверный каталог через NFS и установит загрузочный файл в качестве конфигурационного. При отсутствии сведений о загрузочном файле система будет искать файл `/kickstart/`*`IP_адрес_хоста`*`-kickstart`.

Другой способ задания конфигурационного файла заключается в указании пути к нему в виде аргумента опции `ks`. Здесь есть несколько возможных вариантов. Команда

```
boot: linux ks=http:сервер:/path
```

заставляет утилиту Kickstart загрузить конфигурационный файл по протоколу HTTP, а не через NFS. Аргумент `ks=floppy` заставляет утилиту искать файл `ks.cfg` на локальном гибком диске.

Чтобы вообще не использовать загрузочный носитель, вам нужно научиться работать с протоколом PXE. Подробно о нем речь шла в начале этой главы.

AutoYaST: автоматизированный инсталлятор SUSE

YaST2 — это универсальная утилита инсталляции и конфигурирования SUSE. Она имеет привлекательный графический интерфейс и очень удобна в использовании при инсталляции одной системы. В старых версиях SUSE автоматизация разрешалась с использованием обычной утилиты YaST, однако конечный результат устраивал далеко не всех пользователей. С выходом версии SUSE 8 появилась утилита AutoYaST, которая автоматизировала процесс инсталляции SUSE. Это самое функциональное автоматизированное инсталляционное программное обеспечение из всех дистрибутивов, описанных в этой книге. Подробную документацию можно загрузить с адреса `www.SUSE.com/~ug`.

SUSE разбивает процесс автоматической инсталляции на три фазы: подготовка, инсталляция, конфигурирование. Первоначальная подготовка выполняется с помощью модуля AutoYaST:

```
$ /sbin/yast2 autoyast
```

Этот модуль помогает указать детали требуемой установки. Результатом выполнения этого модуля является управляющий XML-файл, который сообщает инсталлятору о способе конфигурирования системы SUSE; структура файла описана в интерактивном документе, упомянутом выше.

Процесс конфигурирования можно ускорить. Модуль AutoYaST может читать конфигурационные файлы Red Hat Kickstart, помогая модернизировать "устаревшие" системы. Если вам понадобится продублировать конфигурацию на компьютере, на котором вы работаете в настоящий момент, то на этот случай тоже есть вариант автоматизации.

Чтобы произвести инсталляцию, вам понадобится следующее:

- сервер DHCP с той же маской подсети, что и компьютер, на котором вы планируете произвести инсталляцию;
- инсталляционный сервер SUSE или хранилище пакетов;
- сервер, на котором будет размещена информация о конфигурации, необходимая для инсталляции.

Последний из этих серверов может предоставить конфигурационные файлы по одному из протоколов, предлагаемых вам на выбор: HTTP, NFS или TFTP.

При обычной установке нужно создать управляющий файл для каждого компьютера, на котором вы планируете произвести инсталляцию. AutoYaST использует IP-адрес клиента, чтобы определить, какой управляющий файл следует использовать. Этот подход не будет эффективным, если нужно произвести инсталляцию на ряде компьютеров, немного отличающихся друг от друга.

С помощью системы правил можно создавать более сложные инсталляции. На основании таких характеристик системы, как размер диска, идентификатор компьютера или доступность PCMCIA, разные управляющие файлы подгоняются под искомую систему. Содержимое всех выбранных управляющих файлов объединяется, причем в случае возникновения конфликтов содержимое последнего управляющего файла перекрывает содержимое предыдущего файла. (Управляющий файл не должен определять все аспекты конфигурации системы, поэтому такое объединение имеет смысл.)

Управляющие файлы могут также определять "классы" компьютеров на основе их имен или диапазонов IP-адресов. Каждый класс может иметь связанный с ним вспомогательный управляющий файл. Компьютеры могут не относиться к классам, могут относиться к одному классу или к множеству классов, а их конфигурации будут включать содержимое всех управляющих файлов соответствующего класса.

Благодаря объединению содержимого множества управляющих файлов, структура AutoYaST позволяет определять сложные установки с минимальной избыточностью. Человеку трудно воспринимать то, что написано в управляющем XML-файле, зато процесс может с легкостью не только читать эти файлы, но и редактировать их с помощью обычных доступных средств обработки XML-данных.

Инсталлятор Debian и Ubuntu

Инсталлятор Debian (он называется `debian-installer`) был полностью переработан для "сержантского" выпуска. Сейчас он работает без каких-либо проблем. Он используется также в качестве инсталлятора Ubuntu, хотя в Ubuntu имена всех команд напоминают команды Debian (на наш, взгляд, это верный ход).

Новая система имеет модульное строение и написана в комбинации со сценариями оболочки C. Ее можно запускать с двух дискет на компьютерах, на которых установлено как минимум 32 Мбайт оперативной памяти. Она может извлекать все файлы, необходимые для инсталляции, прямо из сети, включая список требуемых файлов.

Все интерактивные части инсталлятора Debian используют утилиту `cdebconf`, чтобы решить, какие вопросы нужно ставить и какие ответы использовать по умолчанию. Предоставив утилите `cdebconf` базу данных с заранее сформулированными вопросами, вы можете полностью автоматизировать работу инсталлятора. Вы можете или сгенерировать базу данных вручную (она представляет собой обычный текстовый файл), или можете выполнить интерактивную инсталляцию на тестовой системе, а затем передать свои ответы утилите `cdebconf` с помощью следующих команд:

```
# debconf-get-selections --installer > config.cfg
# debconf-get-selections >> config.cfg
```

Создайте конфигурационный файл, который будет доступен по сети, и передайте его в ядро во время инсталляции с помощью следующего аргумента ядра:

```
preseed/url=http://компьютер/путь_к_preseed
```

Если вам необходимо развернуть систему Debian автоматически, попробуйте использовать пакет `cdebootstrap`. Он основан на том же коде, что и инсталлятор Debian, и позволяет создавать образ инсталляции в подкаталоге на локальном диске, извлекая необходимые пакеты прямо из хранилища Debian. Это средство может оказаться очень удобным для использования в автоматизированных сценариях установки.

В ранних выпусках Debian инсталлятор являлся источником постоянных жалоб. При необходимости многие системные администраторы реализовывали свои собственные системы инсталляции и конфигурации, многие из которых можно найти по адресу

```
www.linuxmafia.com/faq/Debian/installers.html
```

Другой альтернативой стандартным средствам Debian является пакет, называемый FAI (Fully Automatic Installation — полностью автоматизированная инсталляция). Домашняя страница этого пакета находится по адресу `www.informatik.uni-koeln.de/fai`.

Эти альтернативные системы сейчас нужны меньше, чем в прошлом, однако они попрежнему заслуживают внимания, если вы решите отказаться от способов, предлагаемых стандартным инсталлятором.

Несмотря на то что этот дистрибутив не является "родственником" Red Hat, Ubuntu поддерживает в своем базовом инсталляторе совместимость с управляющими файлами Kickstart. Для создания этих файлов используется утилита `system-config-kickstart`. Функции Kickstart в системе Ubuntu еще не готовы на 100%, однако их можно использовать, не опасаясь каких-либо последствий. В системе Debian совместимость с Kickstart пока что не поддерживается.

Инсталляция из главной системы

Если вы вынуждены не использовать способы инсталляции, которые предлагает стандартный инсталлятор вашего дистрибутива, выясните, можно ли в вашей организации просто дублировать дистрибутив с одного компьютера на другой. Дублирование может быть выполнено быстрее традиционной инсталляции, при этом гарантируется соблюдение единообразия инсталляции.

Может быть, вам подойдет описанный ниже сценарий действий.

- Инсталлируйте и сконфигурируйте главную систему, используя или обычный процесс инсталляции дистрибутива, или сценарий, такой как `debootstrap`, принадлежащий Debian.
- Загрузите компьютер, на который вы хотите перенести дистрибутив, с помощью загрузочного компакт-диска (например, Knoppix[2]) или с использованием PXE.
- Произведите разбиение диска на разделы с помощью утилиты `cfdisk` в интерактивном режиме или с помощью сценария `sfdisk`.
- Создайте файловые системы, инициализируйте пространство для свопинга и смонтируйте все в каталог `/mnt`.
- Продублируйте инсталляцию главной системы на только что размеченном диске с помощью утилит `ssh` или `rsync` (см. раздел 17.3). Не забудьте исключить каталоги `/tmp`, `/proc` и `/sys`.
- Отредактируйте файлы, требующие настройки (например, `/etc/modules` или `/etc/fstab`), или, что еще лучше, автоматизируйте этот процесс.
- Сделайте новый компьютер загружаемым, инсталлировав GRUB или LILO в главную загрузочную запись диска. Более подробно об этом написано в разделе 2.2.

[2] Knoppix — это версия Debian, которая запускается непосредственно с загрузочного компакт-диска (CD или DVD); инсталляция с жесткого диска в этом случае не производится. Этот вариант идеально подходит во время инсталляции, восстановления или во время действий, связанных с организацией защиты. Все подробности можно узнать на сайте `www.knoppix.org`. Сейчас многие другие дистрибутивы (тот же Ubuntu) тоже могут загружаться с компакт-диска.

11.2. Бездисковые клиенты

В 80-х гг. прошлого века, когда жесткие диски стоили очень дорого, многие организации развертывали рабочие станции, не имеющие локальных жестких дисков. Такие компьютеры, называемые "бездисковыми клиентами", монтировали свои файловые системы на сетевом сервере. В наши дни диски стоят недорого, и уже практически никто не использует бездисковые клиенты. Теперь эта идея кажется такой же старомодной, как и использование матричного принтера.

Однако не будем торопиться с выводами. Бездисковые операции по-прежнему допустимы, и для некоторых типов инсталляций они являются актуальными. Ниже перечислены некоторые преимущества этого подхода.

- В бездисковой установке все связанные между собой данные хранятся на сервере. Бездисковые клиенты никогда не окажутся изолированными в том смысле, что информация об их конфигурации станет недоступной. Как результат, администратору будет проще наладить работу вышедшего из строя компьютера.
- Объем информации о локальной конфигурации сводится к минимуму. Практически все операции, связанные с администрированием, не требует физического присутствия перед компьютером-клиентом.
- Настройку новых клиентов можно производить на сервере до того, как они станут работать, или даже до того, как они будут физически подключены. Поведение клиентов во многом подчиняется логическим законам, а не физическим.
- Важные изменения в конфигурации клиента можно выполнять во время перезагрузки. Обновленное корневое дерево может находиться вблизи активного дерева, ожидая подходящего момента для активизации. Благодаря простому программному обеспечению и модернизации дистрибутива, вы можете обеспечить более быстрое и надежное обслуживание компьютеров.
- Для многих приложений дорогой и сложный RAID-массив, организованный в качестве основы сервера, может работать быстрее локальных дисков, даже с учетом сетевых задержек. Более того, сетевой сервер считается более надежным. Локальные диски могут выйти из строя, а использование сервера с резервированием может защитить работу целой сети бездисковых клиентов.
- Неправильно утверждать, что бездисковые клиенты не могут содержать диски. При желании можно использовать локальные диски, чтобы выделить пространство для подкачки или кэша. В наши дни вопрос стоит не в том, чтобы клиенты не имели дисков, а в том, чтобы клиенты не использовали эти диски для хранения данных, требующих внимания со стороны администратора.
- Успешная работа многих приложений зависит от возможностей процессора и практически не зависит от скоростных характеристик диска. Другие приложения, которые потенциально могут нагружать работой диски пользователей, можно переделать так, чтобы их работа ограничивалась нагрузками на процессор и оперативную память. К этой категории можно отнести Web-серверы, серверы баз данных и серверы для сетевых протоколов.

В бездисковой среде клиенты обычно имеют небольшие корневые файловые системы, выделенные для эксклюзивного использования клиентами. Клиенты могут совместно использовать каталог `/usr` (в режиме чтения), так как он не содержит никакой ин-

формации, связанной с работой компьютера, и в него никогда не производится запись во время выполнения обычных операций.

Red Hat Enterprise Linux — единственный дистрибутив Linux, который имеет стандартизированную систему настройки бездисковых клиентов. С настройкой бездисковых клиентов можно справиться самому, даже без профессиональной поддержки со стороны поставщика дистрибутива. Во многих системах реализован механизм загрузки терминалов X Window по сети. Этим механизмом можно воспользоваться как отправной точкой для загрузки рабочих станций по сети.

Если ваш дистрибутив не содержит инструкций по настройке бездисковых клиентов, скопируйте сначала корневую файловую систему с только что инсталлированного компьютера на сервер — она будет являться основой для клонирования новых компьютеров. Определитесь с тем, каким образом будет производиться загрузка ваших систем по сети с использованием корневого каталога NFS. Разрешите всем клиентам совместно использовать части дерева файловой системы компьютера (например, `/usr`) в режиме только чтения.

Клонируя корневой клиент, вам, возможно, придется настроить следующее:

- таблицу монтирования;
- поддержку аппаратных устройств (если ваша сеть не является однородной);
- конфигурацию сети (необязательно);
- сценарии запуска (необязательно).

Компьютерная аппаратура часто продается вместе с дешевыми дисками IDE. Их следует использовать для выделения пространства для свопинга и промежуточных (черновых) разделов.

Для бездисковых узлов сегодня можно выгодно использовать недорогие модули оперативной памяти. Существует программа, называемая RAM disk (ОЗУ-диск), которая позволяет выделить в оперативной памяти место, которое система будет использовать как обычный жесткий диск. Попробуйте использовать файловую систему, построенную на модулях ОЗУ, в качестве резервного места для хранения каталогов `/tmp` и `/dev` (или в качестве кэша для часто используемых файлов, которые будут предварительно загружаться во время загрузки системы)[3]. Если дерево `/dev` будет храниться в ОЗУ, то это позволит значительно повысить производительность, так как программы, которые "любят" размещаться в дереве `/dev`, не будут испытывать проблем с задержками в NFS.

О технологии RAM disk будет рассказано в главе 25.

11.3. Управление пакетами

Все дистрибутивы Linux включают какую-нибудь систему управления пакетами, которая помогает управлять конфигурацией. Традиционно пакеты использовались для распространения программного обеспечения, однако их можно применять также для распространения конфигурационных файлов и административных данных. Пакеты имеют ряд преимуществ по сравнению с неструктурированными архивами `.tar.gz`. Наверное, важнее всего то, что процесс инсталляции пакета протекает как одна транзакция: при возникновении ошибки инсталляцию можно прервать или повторить.

[3] Однако следует иметь в виду, что каталог `/dev` уже может находиться в файловой системе ядра.

Поставщики систем UNIX часто имеют специальные процедуры распространения "заплат" среди своих заказчиков, а дистрибьюторы Linux пользуются преимуществами своих стандартных средств управления пакетами. Чтобы выпустить "заплату", дистрибьютор просто выпускает и модернизирует пакет. Когда пакет инсталлируется на компьютере заказчика, новая версия пакета заменяет старую.

Пакетные инсталляторы обычно знают о конфигурационных файлах и, как правило, не перезаписывают локальные настройки, выполненные системным администратором. Они или создают резервные копии изменяемых файлов, или предлагают образцы конфигурационных файлов под другими именами (например, `pkg.conf.rpmnew`). Если после инсталляции пакета в системе обнаруживаются нарушения, то, теоретически, можно все восстановить в прежнее состояние. Естественно, то, что написано в теории, не всегда проходит на практике, поэтому делать это в производственной системе нужно лишь после проверки установки на тестовой системе.

Системы управления пакетами "понимают" внутрипакетные зависимости, что позволяет разработчикам пакетов быть уверенными в правильной инсталляции не только самих приложений, но и всех библиотек и вспомогательных файлов, от которых они зависят. В ходе инсталляции пакета могут также выполняться различные сценарии (это, наверное, и является причиной большинства известных случаев, когда после деинсталляции пакета система не возвращается в первоначальное состояние).

С помощью пакетов удобно распространять изменения локальных настроек. Можно легко создать пакет, который в процессе инсталляции соберет сведения о компьютере (или получит их из локальной базы данных) и на их основе модифицирует конфигурационные файлы. В виде пакетов можно рассылать локальное ПО вместе со вспомогательными файлами, и даже сторонние приложения, первоначально имевшие другой формат. Пакетам можно присваивать версии, с тем чтобы при инсталляции новой версии пакета автоматически обновлялись зависящие от него компоненты.

Можно также использовать механизм зависимостей для создания групп пакетов. Например, можно создать такой пакет, который сам по себе ничего не инсталлирует, но от него зависит много других "заплат". Появление новой версии пакета приводит к автоматической установке всех этих "заплат" за один раз.

Доступные системы управления пакетами

Широко распространены два формата пакетов. В Red Hat, Fedora и SUSE и большинстве других дистрибутивов применяется RPM (Red Hat Package Manager — диспетчер пакетов Red Hat). В Debian и Ubuntu используются пакеты отдельного формата `.deb`. Оба формата функционально идентичны.

Преобразование из формата в формат можно произвести без каких-либо трудностей с помощью утилиты `alien` (`kitenet.net/programs/alien`). Ей ничего не известно о программах данного пакета, поэтому если содержимое еще не совместимо с вашим дистрибутивом, `alien` не поможет. Вообще, лучше всего использовать "родной" механизм упаковки, используемый в вашем дистрибутиве.

Системы упаковки RPM и `.deb` теперь работают в виде двухуровневых средств управления конфигурацией. На нижнем уровне находятся средства, которые инсталлируют, деинсталлируют и запрашивают пакеты: `rpm` для RPM и `dpkg` для `.deb`.

Над этими командами находятся системы, которые знают, как нужно производить поиск пакетов в Internet, анализировать зависимости между пакетами и модернизировать все пакеты в системе. Основными конкурентами на этом уровне являются `yum`

(работает с системой RPM), Red Hat Network (работает для Red Hat Linux и использует RPM) и Advanced Package Tool (APT, работает для Debian; первоначально она была создана для работы с пакетами `.deb`, а сейчас она может работать также с пакетами RPM).

Следующие две страницы будут посвящены обзору низкоуровневых команд `rpm` и `dpkg`. В разделе 11.4 мы поговорим о комплексных системах обновления.

`rpm`: управление пакетами RPM

Команда `rpm` инсталлирует, проверяет и запрашивает состояние пакетов. Когда-то она еще и создавала пакеты, однако сейчас эта функция отведена команде `rpmbuild`. Тем не менее, опции `rpm` по-прежнему имеют сложные взаимодействия и вместе могут использоваться только в некоторых комбинациях. Воспринимать утилиту `rpm` нужно так, будто это несколько разных команд с одним и тем же именем.

Режим, который вы выбираете для работы `rpm` (например, `--install` или `--query`), определяет, к каким функциям этой утилиты вы хотите обратиться. `rpm --help` перечислит все опции, разбивая их по режимам. Если вам часто приходится иметь дело с пакетами RPM, вам нужно будет внимательно прочесть главную страницу.

Обычно используются опции `--install`, `--upgrade` и `--erase` и `--query`. Последняя опция является довольно сложной в том плане, что она служит для включения остальных опций; чтобы изложить определенный вопрос, вам нужно предоставить дополнительный флаг командной строки. Например, команда `rpm --query --all` отображает список всех пакетов, инсталлированных в системе.

Давайте рассмотрим небольшой пример. Предположим, требуется инсталлировать новую версию пакета OpenSSH, поскольку опубликован бюллетень, в котором сообщается о том, что в предыдущей версии выявлена брешь. После того как пакет загружен на локальный компьютер, для его инсталляции достаточно ввести команду `rpm --upgrade`:

```
# rpm --upgrade openssh-2.9p2-12.i386.rpm
error: failed dependencies:
openssh = 2.9p2-7 is needed by openssh-askpass-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-askpass-gnome-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-clients-2.9p2-7
openssh = 2.9p2-7 is needed by openssh-server-2.9p2-7
```

Гм... Похоже, не все так просто! Как видите, инсталлированная в настоящий момент версия 2.9p2-7 связана с рядом других пакетов. Команда `rpm` не позволяет обновить пакет OpenSSH, так как это изменение затрагивает другие пакеты. Этот тип конфликта встречается постоянно, поэтому лучше разрабатывать такие системы, как APT и `yum`. В реальном мире мы вряд ли пытались бы распутывать зависимости вручную, однако в качестве примера мы сделаем это для утилиты `rpm`.

Можно прибегнуть к принудительному обновлению с помощью опции `--force`, но это вряд ли оправдано. Ведь это не Windows; информация о зависимостях присутствует здесь для того, чтобы сэкономить ваше время и избавить вас от лишних проблем, а не для того, чтобы запутать вас. Для любого системного администратора нет ничего хуже, чем нарушение работы SSH в дистанционной системе.

Вместо этого мы будем использовать обновленные версии пакетов зависимостей. Если бы мы были находчивыми, то перед модернизацией мы могли бы определить, что остальные пакеты зависят от OpenSSH:

```
# rpm --query --whatrequires openssh
openssh-askpass-2.9p2-7
openssh-askpass-gnome-2.9p2-7
```

```
openssh-clients-2.9p2-7
openssh-server-2.9p2-7
```

Теперь предположим, что обновленные копии необходимых пакетов получены. Можно инсталлировать их последовательно, но команда `rpm` все берет на себя. Достаточно указать список пакетов в командной строке, и команда `rpm` отсортирует их в соответствии с имеющимися зависимостями.

```
# rpm --upgrade openssh-*
```

Проверим результат:

```
# rpm --query openssh
openssh-2.9p2-12
```

Все получилось!

Обратите внимание на тот факт, что `rpm` понимает, о каком пакете идет речь, даже если не указывать полное имя или версию пакета.

dpkg: управление пакетами Debian

В Debian аналогом команды `rpm` является команда `dpkg`. К полезным ее опциям относятся `--install`, `--remove`, а `-l` перечисляет пакеты, инсталлированные в системе. Обратите внимание на то, что команда `dpkg --install`, выполненная в отношении пакета, уже находящегося в системе, перед инсталляцией удаляет предыдущую версию пакета.

Предположим, группа, занимающаяся вопросами безопасности Debian, выпустила "заплату" к редактору `nvi`. После загрузки "заплаты" нужно выполнить команду `dpkg` для ее установки. Из показанного ниже примера видно, что эта команда гораздо более многословна, чем `rpm`, и сообщает о том, что именно она делает.

```
# dpkg --install ./nvi_1.79-16a.1_i386.deb
(Reading database ... 24368 files and directories currently installed.)
Preparing to replace nvi 1.79-14 (using ./nvi_1.79-16a.1_i386.deb) ...
Unpacking replacement nvi ...
Setting up nvi (1.79-16a.1) ...
Checking available versions of ex, updating links in /etc/alternatives ...
(You may modify the symlinks there yourself if desired - see 'man ln'.)
Leaving ex (/usr/bin/ex) pointing to /usr/bin/nex.
Leaving ex.1.gz (/usr/share/man/man1/ex.1.gz) pointing to
  /usr/share/man/man1/nex.1.gz.
...
```

Теперь можно ввести команду `dpkg -l`, чтобы узнать, все ли прошло нормально. Флаг `-l` допускает наличие шаблона поиска, благодаря чему можно получить информацию только по редактору `nvi`:

```
% dpkg -l nvi
Desired=Unknown/Install/Remove/Purge
| Status=Not/Installed/Config-files/Unpacked/Failed-config/
Half-installed
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err:
 uppercase=bad)
||/ Name         Version         Description
+++-============-===============-==================================
ii  nvi          1.79-16a.1      4.4BSD re-implementation of vi.
```

Инсталляция завершилась успешно.

11.4. Высокоуровневые системы управления пакетами

Системы управления пакетами, такие как APT и `yum`, а также Red Hat Network, ставят перед собой следующие задачи:

- упростить определение местонахождения и загрузку пакетов;
- автоматизировать процесс обновления или модернизации систем;
- способствовать управлению зависимостей между пакетами.

Понятно, что помимо команд на стороне клиентов эти системы должны выполнять множество других функций. Все они требуют, чтобы компании, обеспечивающие поддержку дистрибутивов, организовывали свои предложения в согласованном порядке, чтобы клиенты могли иметь доступ к их ПО осознанно.

Так как ни один поставщик не в состоянии предложить программы для Linux "на любой вкус", любая система допускает существование множества хранилищ программного обеспечения. Хранилища могут быть локальными по отношению к вашей сети, поэтому эти системы предлагают первоклассную базу, чтобы вы могли создать собственную систему внутреннего распределения.

Служба Red Hat Network неразрывно связана с дистрибутивом Red Hat Enterprise Linux. Это коммерческая служба, пользование которой стоит денег, зато она обладает привлекательным интерфейсом и более мощными возможностями в плане автоматизации, чем APT и `yum`. К сожалению, с точки зрения особенностей работы она является настоящим "черным ящиком". Клиент может ссылаться на хранилища `yum` и APT, и эта возможность позволяет таким дистрибутивам, как CentOS, адаптировать клиентский графический интерфейс под нестандартное использование.

APT документирована гораздо лучше, чем Red Hat Network, и к тому же бесплатна. Она также является более гибкой в плане настройки. APT разрабатывалась для Debian и `dpkg`, но была расширена, и теперь может работать с RPM. Доступными являются все версии, работающие со всеми нашими примерами дистрибутивов. На данный момент система APT более всех может походить на универсальный стандарт для распространения программного обеспечения.

`yum` — это аналог APT, предназначенный для RPM. `yum` является также стандартным администратором пакетов для Fedora, хотя может работать в любой системе, основанной на RPM, при условии, что вы сможете указать ей хранилища, имеющие соответствующий формат.

Если провести сравнение, то оказывается, что `yum` предпочтительнее RPM-версии APT, хотя в пользу такого утверждения невозможно найти веские технические доводы. `yum` имеет более глубокие корни в мире RPM, что дает ей гипотетическое преимущество. Будущее APT-RPM было неясным в начале 2005 г., когда ее автор, Густаво Нимейер (Gustavo Niemeyer), отказался от дальнейшей ее разработки, переключившись на работу над более сложной системой (Smart Package Manager, `www.labix.org`; эту систему нельзя назвать передовой, однако считается, что за ней стоит будущее). Позже разработку APT-RPM возобновил Пану Матилайнен (Panu Matilainen), поэтому сейчас проект снова находится на стадии активной разработки.

Нам нравится APT, и мы считаем, что это разумный выбор, если нужно настроить собственную автоматизированную сеть распределения пакетов, независимо от исполь-

зуемого в данный момент дистрибутива. Более подробно об этом можно прочитать в разделе "Создание внутреннего сервера APT" далее в этой главе.

SUSE — это неудачная система среди систем управления пакетами. Она использует пакеты RPM, однако в предыдущих версиях для выполнения обновлений системы поддерживалась только утилита YaST Online Update, принадлежащая SUSE. С недавних пор SUSE стала частично поддерживать (эта поддержка оставляет желать лучшего) `yum`, APT-RPM и агент ZENworks Linux Management (разработка компании Novell), причем ZENworks принят как основной администратор обновлений. (Интерфейс командной строки можно посмотреть в `rug`.)

ZENworks является частью большой линейки продуктов, за счет которых Novell старается захватить лидирующую позицию в области межплатформенного управления конфигурациями. Будет ли ZENworks оптимальным вариантом для вас? Может, это и так, если вы являетесь дистрибьютором продукции Novell и хотите получить оплаченную поддержку. Организации, которые желают, чтобы их управление пакетами было бесплатным и относительно аполитичным, могут использовать систему Smart Package Manager, о которой было упомянуто выше.

Хранилища пакетов

Дистрибьюторы Linux поддерживают хранилища программного обеспечения, которые работают рука об руку с выбранными ими системами управления пакетами. Конфигурация, выбираемая по умолчанию для системы управления пакетами, обычно указывает на один или несколько хорошо известных Web- или FTP-серверов, находящихся под управлением дистрибьюторов.

Однако из этого факта нельзя сразу понять, что может содержаться в таких хранилищах. Должны ли они включать только наборы пакетов, принятых в качестве официальных, основных версий? Официальные версии плюс текущие обновления системы защиты? Современные версии всех пакетов, которые существовали в официальных выпусках? Полезные программы сторонних производителей, которые официально не поддерживаются дистрибьютором? Исходный код? Бинарные файлы для многочисленных архитектур аппаратных средств? Если вы запускаете `apt-get upgrade` или `yum upgrade`, чтобы обновить систему, то что именно под этим подразумевается?

Вообще, системы управления пакетами должны отвечать на все эти вопросы и должны облегчать организациям выбор специфических профилей, которые они хотят определить как их "мир программного обеспечения". Следующие концепции помогут структурировать этот процесс.

- "Выпуск" (release) — это самосогласованное отображение "вселенной" пакета. Прежде чем наступила эпоха Internet, именованные выпуски операционных систем были более или менее постоянными и были связаны с одним определенным моментом времени; "заплаты" системы защиты создавались отдельно. В наши дни выпуск представляет собой более расплывчатое понятие. Выпуски выходят в то время, когда производится обновление пакетов. Некоторые выпуски, такие как Red Hat Enterprise Linux, предназначены специально для того, чтобы задержать выходы новых версий; по умолчанию в них включаются только обновления защиты. Остальные выпуски, такие как бета-версии, меняются часто и существенно. Однако во всех случаях выпуск является базовой линией, трендом, или той мерой "до которой я хочу обновить свою систему".

- "Компонент" (component) — подборка программ в рамках выпуска. Дистрибутивы имеют между собой разные отличия, однако общим является отличие между основным программным обеспечением, предлагаемым дистрибьютором, и дополнительным программным обеспечением, предлагаемым энтузиастами. Другое отличие, обычное для мира Linux, кроется между свободными частями открытого исходного кода выпуска и частями, связанными с некоторым соглашением о коммерческом использовании.

  Отдельного внимания со стороны администратора заслуживают минимально активные компоненты, которые включают только исправления в системе защиты. Некоторые выпуски позволяют комбинировать компонент защиты с постоянным базовым компонентом для создания относительно стабильной версии дистрибутива.
- "Архитектура" (architecture) — это специфический класс аппаратных средств (оборудования). Предполагается, что компьютеры, относящиеся к некоторому классу архитектуры, будут иметь одинаковые характеристики, позволяющие запускать на этих компьютерах одинаковые исполняемые файлы. Архитектуры являются специфическими экземплярами выпусков (например, "Fedora Core 5 for the i386 architecture"). Так как компоненты являются подразделениями выпусков, то для каждого из них существует соответствующий экземпляр, характерный для данной архитектуры.
- Индивидуальные пакеты являются элементами, составляющими компоненты, а, следовательно, и выпуски. Пакеты обычно являются специфическими для архитектуры, и выходят в виде версии, не зависящей от главного выпуска и остальных пакетов. Соответствие между пакетами и выпусками является неявным в том плане, как производится настройка сетевого хранилища.

Существование компонентов, которые не поддерживаются дистрибьютором (например, "contrib." для Debian или "extras" для Fedora), поднимает вопрос о том, как эти компоненты относятся к основному выпуску ОС. Можно ли о них говорить как о настоящих "компонентах" специфического выпуска, или же они представляют собой нечто другое? С точки зрения управления пакетами ответ прост: "extras" — это настоящий компонент. Они связаны со специфическим выпуском и выходят в паре с ним. Разделение управления интересно с точки зрения администрирования, но не влияет на системы управления пакетами.

Служба Red Hat Network

После того как система Red Hat перестала входить в группу "потребительских товаров" Linux, Red Hat Network стала платформой для управления системой для Red Hat Enterprise Linux. Подписываясь на нее, вы приобретаете право доступа к Red Hat Network. В простейшем виде служба Red Hat Network представляет собой популярный Web-портал и список рассылки. В этом смысле она мало чем отличается от тех служб уведомления о "заплатах", которые уже много лет эксплуатируются различными поставщиками UNIX-систем. Дополнительные возможности открываются лишь при условии, что вы готовы за них заплатить. Информацию и текущие расценки можно узнать на сайте `rhn.redhat.com`.

Служба Red Hat Network предлагает удобный интерфейс загрузки новых пакетов (можно также работать в режиме командной строки). Она даже позволяет загружать и инсталлировать пакеты без вмешательства человека. После того как вы зарегистриру-

тесь, система будет получать все необходимые "заплаты" в автоматическом режиме. Недостатком такого подхода является то, что решения об обновлении системы принимаются за вас. Определите для себя, в какой степени следует доверять разработчикам Red Hat (а также тех программ, которые они предлагают в виде пакетов).

Разумным компромиссом будет выделение одного из компьютеров в качестве сервера автоматических обновлений. Периодически можно создавать образы серверной системы и проверять, в какой степени они подходят для внутреннего распространения. Однако перед тем как делать это, нужно обязательно ознакомиться с лицензионными условиями Red Hat (их можно найти по адресу `www.redhat.com/licenses`). Вас может удивить тот факт, что Red Hat заявляет о праве собственности на программное обеспечение с открытым исходным кодом, распространяемое через Red Hat Network, не говоря уже о том, что вы должны согласиться с тем, что Red Hat будет проверять ваши системы по ее усмотрению.

APT: усовершенствованное средство управления пакетами

APT (Advanced Packaging Tool) — это одна из зрелых систем управления пакетами. С помощью всего лишь одной команды `apt-get` можно модернизировать всю систему и программное обеспечение; также с ее помощью (как и с помощью Red Hat Network) можно постоянно сохранять ваши "коробчные" версии современными без вмешательства со стороны человека.

Поскольку APT разрабатывалась для Debian, она изначально поддерживала только пакеты формата `.deb`. Но сейчас появилась версия, поддерживающая пакеты RPM. Эта версия называется APT-RPM и доступна на сайте `apt-rpm.org`. Система `yum` (описание см. в разделе "yum: управление выпусками для RPM" далее в этой главе) предлагает функции, похожие на RPM, и напрямую поддерживается выпусками Fedora. При выборе между `yum` и APT-RPM на первый план выходит неоднородность компьютеров в вашей организации и предпочтительные настройки вашего дистрибутива, а не технические различия между системами. Начинать нужно с того, что имеет лучшую поддержку и проще используется.

Первое правило, которого следует придерживаться при использовании утилиты `apt-get` в Debian (это касается всех средств управления пакетами этой системы), требует игнорировать утилиту `dselect`, являющуюся клиентской надстройкой системы. Речь не идет о том, что утилита `dselect` плохая, просто ее интерфейс неудачен. В документации к Debian вы найдете рекомендации в пользу использования `dselect`, однако их не следует принимать во внимание.

В случае типичной инсталляции Debian, которая загружается с одного из стандартных "зеркал", узнать список доступных пакетов можно на Web-сайте `packages.debian.org` или `packages.ubuntu.com`. Оба дистрибутива имеют удобную поисковую систему. Если же создается внутренний сервер APT (см. раздел "Создание внутреннего сервера APT" в этой главе), то, конечно, администратор сам знает, какие пакеты доступны, и может сформировать их список.

Дистрибутивы обычно имеют пустые пакеты, которые существуют для того, чтобы формировать списки зависимых пакетов. Утилита `apt-get` автоматически загружает и инсталлирует зависимые пакеты, благодаря чему можно легко устанавливать или обновлять блоки пакетов. Например, инсталлируя пакет `gnome-desktop-environment`,

можно быть уверенным в том, что установлено все необходимое для пользовательского интерфейса GNOME.

Если файл `sources.list` настроен, а имя необходимого пакета известно, то осталось лишь выполнить команду `apt-get update`, чтобы обновить информацию о пакетах. После этого нужно ввести команду `apt-get install` *`имя_пакета`*, которая, собственно, и осуществит инсталляцию пакета. Эта же команда обновит пакет, если он уже инсталлирован.

Предположим, требуется установить новую версию пакета `sudo`, в котором устранена очередная брешь. Сначала не помешает выполнить команду `apt-get update`:

```
% sudo apt-get update
Get:1 http://http.us.debian.org potato/main Packages [824kB]
Get:2 http://non-us.debian.org potato/non-US/main Release [102B]
...
```

Теперь можно приступить к получению пакета. Обратите внимание на то, что мы используем команду `sudo` в процессе инсталляции нового пакета `sudo` — утилита `apt-get` способна обновлять даже те пакеты, которые в настоящий момент используются.

```
% sudo apt-get install sudo
Reading Package Lists... Done
Building Dependency Tree... Done
1 packages upgraded, 0 newly installed, 0 to remove and 191 not upgraded.
Need to get 0B/122kB of archives. After unpacking 131kB will be used.
(Reading database ... 24359 files and directories currently installed.)
Preparing to replace sudo 1.6.1-1 (using .../sudo_1.6.2p2-2_i386.deb) ...
Unpacking replacement sudo ...
Setting up sudo (1.6.2p2-2) ...
Installing new version of config file /etc/pam.d/sudo ...
```

Конфигурирование

Сконфигурировать утилиту `apt-get` несложно. Хорошие инструкции можно найти в документе *APT HOWTO* по адресу

```
http://www.debian.org/doc/manuals/apt-howto
```

Самый важный конфигурационный файл утилиты называется `/etc/apt/sources.list`. В нем сообщается, где искать пакеты. В каждой строке файла указывается следующее.

- Тип пакета. В настоящее время это `deb` или `deb-src` для пакетов в стиле Debian или `rpm` или `rpm-src` для RPM.
- URL-адрес файла, компакт-диска, сервера HTTP или FTP, где находятся пакеты.
- "Дистрибутив" (на самом деле — название выпуска), если нужно работать с несколькими версиями пакета. Дистрибьюторы используют его для главных выпусков, однако вы можете использовать его для других целей, если вам нужны внутренние системы распространения.
- Необязательный список компонентов, т.е. категорий пакетов в рамках дистрибутива.

Стандартная конфигурация вполне приемлема, если только не нужно создавать собственное хранилище пакетов. При наличии быстродействующего сетевого соединения можно закомментировать строки, ссылающиеся на инсталляционный компакт-диск

дистрибутива. Если требуется загружать исходный код, снимите комментарии с тех строк, где тип пакета равен `deb-src` или `rpm-src`.

В процессе редактирования файла можно поменять адрес "зеркала", задав более близко расположенный сервер. Полный список "зеркал" находится по адресу

```
http://www.debian.org/misc/README.mirrors
```

Ubuntu поддерживает похожий список (`wiki.ubuntu.com/Archive`).

Ситуация еще больше упрощается благодаря утилите `netselect-apt`, которая автоматически генерирует файл `sources.list`. Она выбирает ближайшее "зеркало" на основании времени прохождения эхо-запроса команды `ping`. Эта утилита является частью пакета `netselect`, который также можно найти на дистрибутивных серверах. (Готовый вариант `netselect-apt` отчасти привязан к системе "зеркал" Debian, однако большинство пакетов будет работать также и с Ubuntu.)

Убедитесь, что в качестве одного из источников указан сервер `security.debian.org` или `security-ubuntu.com`, чтобы можно было оперативно получать самые последние "заплаты", связанные с безопасностью.

Пример файла /etc/apt/sources.list

В показанном ниже примере стабильная версия дистрибутива загружается с сервера `http.us.debian.org`, а с сервера `non-us.debian.org` загружаются пакеты, созданные за пределами США (первоначально это было необходимо для криптографических пакетов, однако сейчас это уже практически не имеет значения). Мы также добавили сервер `security.debian.org` в качестве источника "заплат", связанных с безопасностью, и наш локальный сервер APT (`debserver`). Последние строки разрешают загрузку исходных кодов:

```
# Общий формат: тип URL-адрес дистрибутив [компоненты]
deb http://ftp.us.debian.org/debian stable main contrib non-free
deb http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb http://security.debian.org stable/updates main contrib non-free
deb http://local-debian-server/mypackages/ ./
deb-src http://http.us.debian.org/debian stable main contrib non-free
deb-src http://non-us.debian.org/debian-non-US stable/non-US main contrib non-free
deb-src http://security.debian.org stable/updates main contrib non-free
```

Поля *дистрибутив* и *компоненты* помогают утилите `apt-get` ориентироваться в иерархии файловой системы хранилища Debian, которая характеризуется стандартизированным размещением. Корневой дистрибутив может быть помечен как `stable` (самый последний главный дистрибутив), `unstable` (или `testing` — текущий дистрибутив, находящийся в разработке), или может иметь имя определенного выпуска, например: `etch`. Доступными компонентами обычно являются `main`, `contrib` и `non-free`.

В этом примере работа ведется с хранилищем стабильной версии дистрибутива, которая меняется не так часто, как другие версии. Все свежие пакеты включены в дистрибутив Debian `unstable`. "Нестабильный" (unstable) не означает, что пакеты являются нестабильными. Просто состав всего дистрибутива может меняться. Обычно за одну неделю производятся изменения объемом более 100 Мбайт.

Строки в файле `sources.list` рассматриваются по порядку, поэтому теоретически вы можете поместить строки `unstable` и `testing` в конец файла, чтобы дать преимущество стабильным версиям. Однако при таком подходе может возникнуть следующая

проблема: при отслеживании зависимостей APT один нестабильный пакет может "потянуть" за собой обновленные, нестабильные версии всех пакетов, от которых он зависит. Эти пакеты, в свою очередь, могут "потянуть" за собой нестабильные версии всех своих "приятелей", и так далее. Одна ложка дегтя испортит целую бочку меда; в производственных системах лучше не экспериментировать с нестабильными пакетами.

Если вам нужно включить пакет из нестабильного "мира" в вашу производственную среду, используйте для этого т.н. "ретроподдержку" (backport) — часть кода из последней версии программы к более старой версии, перекомпилированного в стабильном выпуске, чтобы добиться согласованности со стабильными библиотеками. Чтобы найти такие части кода, воспользуйтесь поисковой системой APT (`www.apt-get.org`). Многие пакеты в таких портах (в отличие от обычных указателей) можно найти на сайте Норберта Третковски (Norbert Tretkowski) по адресу `www.backports.org`. Коды "ретроподдержки" в этом хранилище имеет высокое качество и минимум внешних зависимостей.

Использование прокси-серверов для кэширования пакетов

Если планируется применять утилиту `apt-get` для большого числа компьютеров, то потребуется локальное кэширование пакетов — загружать копии всех пакетов для каждого компьютера было бы неразумно. Можно также сконфигурировать утилиту на работу через прокси-сервер, если того требуют настройки брандмауэра.

Утилита использует обычные протоколы HTTP и FTP, поэтому подойдет любой существующий прокси-сервер. Утилита проверяет значение переменной среды `http_proxy`; кроме того, прокси-сервер можно задать с помощью такой строки файла `/etc/apt/apt.conf`:

```
Acquire::http::Proxy "http://прокси-сервер:8080/"
```

В качестве альтернативы универсальному прокси-серверу подойдет небольшое приложение `apt-proxy`. Несмотря на название, это не настоящий прокси-сервер, а программа, которая формирует кэш пакетов, загружая их с помощью утилиты `rsync` с сервера APT. Эта утилита доступна по адресу

```
http://sourceforge.net/projects/apt-proxy
```

Создание внутреннего сервера APT

Вместо прокси-сервера можно создать собственный автономный сервер APT и нацелить на него внутренние клиенты. Такая модель позволяет контролировать перечень пакетов, предлагаемых клиентам, легко распространять обновления (достаточно поместить новые версии на сервер), распространять собственные приложения в виде пакетов и, самое главное, создавать собственные версии дистрибутивов.

Поскольку утилита `apt-get` загружает пакеты, пользуясь стандартными протоколами (HTTP и FTP), для организации сервера APT достаточно создать обычный FTP- или Web-сервер, на котором будет размещено соответствующее содержимое[4]. Учитывая широкую поддержку протокола HTTP прокси-серверами и брандмауэрами, этот протокол удобнее всего использовать на сервере APT.

[4] Есть и другие варианты. Например, можно создать сервер NFS, который будет содержать файлы обновления, или подготовить DVD-диск, с которым вы будете обходить системы.

О настройке Web-сервера Apache можно прочитать в разделе 21.2.

Все пакеты на сервере могут находиться в одном каталоге, или это может быть иерархия каталогов, как на "зеркалах" Debian и Ubuntu.

Помимо того, что вы должны предоставить файлы пакетов, нужно создать два сводных файла: `Packages.gz` и `Contents.gz`. В первом из них (он должен быть упакован с помощью `gzip`) содержится список пакетов на сервере и их зависимостей. Команда `apt-get update` проверяет этот список на предмет того, не появились ли дополнения. Второй файл связывает обычные файлы с пакетами, в которых они содержатся. Этот файл не используется самой утилитой `apt-get`. Команда `apt-ftparchive`, входящая в пакет `apt-utils`, автоматически генерирует оба сводных файла.

Далее все просто. Строка вида

```
deb http://debserver/mypackages/ ./
```

в файле `/etc/apt/sources.list` на клиентских компьютерах связывает утилиту `apt-get` с локальным сервером. Выполните на каждом клиенте команду `apt-get update`, после чего можно будет использовать утилиту в обычном режиме.

Если требуется распространять не только бинарные пакеты, но также исходные коды, разместите пакеты исходных кодов на сервере. В отличие от стандарта RPM, где есть специальный формат SRPM для таких пакетов, в Debian пакеты исходных кодов разбиваются на три части: обычный файл `.tar.gz`, необязательный файл `.diff.gz` (используется разработчиками пакетов для демонстрации изменений, которые они внесли в исходную версию кода) и файл `.dsc` с описанием пакета. Соответствующий сводный файл называется `Sources.gz`; он тоже генерируется командой `apt-ftparchive`.

В рассмотренном примере параметр "дистрибутив" не указывался. Если требуется работать с различными внутренними версиями дистрибутива, поместите каждую из них в отдельный подкаталог и поменяйте параметр `./` в приведенной выше строке на имя или номер версии.

Механизм версий очень удобен. Можно создать разные версии дистрибутива с именами "test" и "production", аналогичные выпускам "stable" и "testing" в Debian. Чтобы сопоставить эти имена с определенными выпусками, используйте символические ссылки на сервере. Позже можно будет переопределить содержимое выпусков, просто изменив целевые объекты в ссылках. Например, когда тестовый выпуск будет готов к развертыванию, вы сможете задать ссылку "production" на тот же каталог. Клиенты, синхронизирующиеся с выпуском "production", впоследствии смогут автоматически принимать изменения.

Автоматизация работы утилиты

Утилиту `apt-get` можно запускать по графику с помощью демона `cron`. Даже если пакеты не инсталлируются автоматически, полезно регулярно выполнять команду `apt-get update`, чтобы следить за обновлениями сводных файлов.

Команда `apt-get dist-upgrade` загрузит и инсталлирует новые версии любых пакетов, имеющихся на локальном компьютере. Флаг `dist-upgrade` эквивалентен флагу `upgrade`, но включает более точную обработку зависимостей. `dist-upgrade` может удалить пакеты, которые утилита посчитает несовместимыми с модернизированной системой, так что к этому нужно быть готовым.

Можно даже позволить компьютерам выполнять обновления в автоматическом режиме. Для этого предназначена опция `-yes`, при наличии которой на любой вопрос, задаваемый утилитой `apt-get`, будет выдано оптимистичное "Да!".

Обычно не рекомендуется автоматически загружать обновления непосредственно с "зеркал" дистрибутивов. Другое дело, когда имеются внутренние серверы APT и система управления версиями. Следующий маленький сценарий позволит клиенту поддерживать синхронизацию с сервером APT:

```
#!/bin/sh
apt-get update
apt-get -yes dist-upgrade
```

Этот сценарий может запускаться демоном `cron` каждую ночь. Можно также создать ссылки на него в сценариях запуска системы (см. главу 2), чтобы обновления происходили на этапе начальной загрузки.

Если процедуры обновления запускаются на большом числе компьютеров, необходимо развести их по времени, чтобы не перегрузить сеть. Это можно сделать с помощью небольшого Perl-сценария, приведенного в разделе 17.3.

Если вы не вполне доверяете источнику пакетов, можно автоматически загружать их, но не инсталлировать. Это позволяет делать опция `--download-only` утилиты `apt-get`. Просмотрите полученные пакеты и определите сами, какие из них инсталлировать. Загруженные пакеты находятся в каталоге `/var/cache/apt`, который со временем может серьезно разрастись. Удаляйте неиспользуемые файлы командой `apt-get autoclean`.

Если вы используете официальный стабильный дистрибутив Debian, мы можем порекомендовать выполнять автоматические обновления без резервирования. Изменения в стабильном дистрибутиве обычно ограничиваются обновлениями в системе защиты. Единственная проблема, которая может возникнуть, заключается в том, что вам может не понадобиться автоматическая модернизация при выходе нового главного выпуска. Чтобы эта проблема не возникла, укажите в файле `source.list` явное имя дистрибутива вместо ключевого слова `stable`.

`yum`: управление выпусками для RPM

Система `yum` (Yellowdog Updater, Modified) представляет собой администратор метапакетов, основанный на RPM[5]. Называть `yum` клоном `apt-get` будет, пожалуй, неправильно, однако в плане тематики и реализации она очень похожа на `apt-get`, а на практике — проще и медленнее. `yum` — это официальная система управления пакетами для Fedora; она предварительно устанавливается и на многих других дистрибутивах. При необходимости самую последнюю версию `yum` можно получить по адресу `linux.duke.edu/yum`.

Как и в случае с `apt-get`, команда на стороне сервера (`yum-arch`) компилирует базу данных заголовочной информации из большого набора пакетов (нередко из целого выпуска). После этого база данных заголовков совместно используется пакетами посредством протоколов HTTP или FTP. Клиенты используют команду `yum` для выбора и инсталляции пакетов; `yum` выявляет ограничения зависимостей и выполняет дополнительные действия, необходимые для завершения процесса инсталляции требуемых пакетов. Если запрошенный пакет зависит от других пакетов, `yum` загружает и инсталлирует и эти пакеты.

[5] Yum Fish Bait не следует путать с Live Prey Technology (LPT), `yum3x.com`.

Подобие между apt-get и yum расширяется на опции командной строки, которые понятны им обоим. Например, yum install foo загружает и инсталлирует самую свежую версию пакета foo (и его зависимости, если это необходимо). Однако существует как минимум одно "предательское" отличие: apt-get update обновляет кэш информации о пакетах apt-get, а yum update обновляет каждый пакет в системе (аналогично команде apt-get update). Более того, есть еще команда yum upgrade, которая делает то же, что и yum update, но устаревшими приемами.

yum не рассматривает частичные имена пакетов, если не включить символы универсализации оболочки (такие как * и ?). Например, yum update 'perl*' обновляет все пакеты, имена которых начинаются с "perl". Не забывайте заключать символы универсализации в кавычки, чтобы избежать возникновения ошибок.

В отличие от apt-get, yum во время запуска по умолчанию сверяет информацию о пакетах, хранящуюся в кэше, с содержимым сетевого хранилища. Чтобы отменить этот процесс, используйте опцию -C, в результате чего yum makecache будет обновлять локальный кэш (на это уйдет некоторое время). К сожалению, опции -C недостаточно, чтобы повысить производительность медлительной yum.

Конфигурационным файлом yum является /etc/yum.conf. Он включает общие опции и указатели на хранилища пакетов. Можно активизировать одновременно множество хранилищ; каждое хранилище может быть связано с множеством URL-адресов.

Fedora включает метаданные формата yum в своих стандартных деревьях дистрибутивов, поэтому пользователи Fedora могут просто указать на свои файлы yum.conf на ближайшем "зеркале" и обновить их по своему усмотрению. RHEL в этом отношении не допускает ничего подобного, возможно, чтобы способствовать продажам подписок на Red Hat Network.

11.5. Управление изменениями

Ошибки — неизменный аспект нашей жизни. Следовательно, очень важно отслеживать производимые вами изменения, чтобы в случае возникновения ошибок, вызванных этими изменениями, можно было без труда вернуться к надежной конфигурации. Определение официальных внутренних выпусков позволяет облегчить откат, однако это решение слишком громоздкое. А если нужно просто настроить один или два файла?

В этом разделе мы рассмотрим некоторые распространенные способы управления изменениями на уровне отдельных файлов. Эти способы являются дополнениями к задаче внутреннего управления изменениями и дополняют друг друга — достаточно выбрать ассортимент средств, которые будут совпадать с вашими локальными потребностями и степенью сложности вашей организации.

Создание резервных файлов

Создание резервных копий модифицируемых файлов — это давно устоявшаяся традиция среди системных администраторов. Резервные копии создаются или сценариями локализации, или самими администраторами. Резервирование файлов позволяет вернуться к более ранней конфигурации, однако более важным является то, что вы сможете выполнить команду diff для сравнения текущей и предыдущей версии файла, чтобы выяснить, какие изменения были произведены.

Чтобы создать резервные файлы, лучше всего переместить (с помощью команды mv) исходный файл куда-нибудь под другим именем (например, *имя_файла*.old или

имя_файла.bak), а затем скопировать его обратно, присвоив ему прежнее имя. С помощью опции -p команды cp можно сохранить настройки атрибутов файла. После того как активная версия файла будет обновлена, время его изменения покажет, когда было произведено последнее изменение, а время изменения в резервном файле покажет время предыдущего изменения. (Если команду cp выполнять без опций, оба файла будут иметь одинаковое время изменения.) Также, перемещать исходный файл полезно в том случае, если активный процесс имеет открытую ссылку на данный файл: производимые вами изменения в неактивной копии не вступят в силу до тех пор, пока вы не закроете файл и не откроете его снова.

Если система регулярно резервируется на ленту, то в этом случае лучше перейти к созданию резервных файлов вручную. Восстановление из резервного файла всегда производится быстрее и проще, чем с ленты; более того, резервные копии, созданные вручную, сохраняют дополнительный слой истории.

Формальные системы управления изменениями

Резервные файлы очень полезны, однако самый ощутимый эффект они приносят для небольших организаций. Следующий уровень сложности и устойчивости занимают формальные системы управления изменениями, которые представляют собой пакеты программ, отслеживающих, архивирующих и обеспечивающих доступ к множеству версий файлов. Эти пакеты первоначально использовались при разработке программного обеспечения, однако они будут полезными и для системных администраторов.

Системы управления изменениями позволяют решить несколько проблем. Во-первых, они обеспечивают организованный доступ к отслеживанию истории изменений в файле, благодаря чему любые изменения можно понять исходя из контекста, а также можно восстанавливать ранние версии файлов. Связанные группы файлов могут вместе входить в одну версию, при этом будут учтены все зависимости. Наконец, системы управления изменениями координируют работу множества редакторов, поэтому вряд ли возникнет ситуация, когда чьи-то изменения будут навсегда утеряны[6], и когда несовместимые изменения, производимые в нескольких редакторах, окажутся одновременно активными.

Самой простой и широко используемой системой управления изменениями является RCS (Revision Control System). Она существует уже несколько десятков лет и предварительно инсталлируется на многих системах.

Другим вариантом является система с открытым исходным кодом, называемая CVS (Concurrent Versions System — система параллельных версий), которая, по сравнению с RCS, обладает дополнительными функциями. Она поддерживает распределенную модель (для использования с дистанционным сервером) и более совершенную поддержку множества разработчиков. В некоторых организациях эта система используется для задач администрирования; в частности, очень полезной может быть система клиент/сервер. К сожалению, концепции CVS имеют некоторые недостатки.

Более свежим (но по-прежнему тестируемым) игроком на арене открытого кода является Subversion — система, которая обладает всеми преимуществами CVS, но по умолчанию имеет более удобное поведение. Ее основным недостатком для системных адми-

[6] Предположим, например, что системные администраторы Алиса и Боб оба редактируют один и тот же файл, и каждый из них вносит в файл некоторые изменения. Алиса первая сохранила файл. Когда Боб сохранил свою копию файла, он перезаписал версию Алисы. Если Алиса закроет редактор, ее изменения будут полностью утрачены и не смогут быть восстановлены.

нистраторов является то, что модель проектов сосредоточена на каталогах. И все же, эта система очень хороша и рекомендуется всем администраторам.

В течение нескольких последних лет наблюдался бум в области систем управления версиями с открытым исходным кодом, причем доступные варианты имели между собой серьезные отличия. Среди новых систем лидирующими являются Monotone, Darcs, Arch и Bazaar-NG. Все они по-своему интересны, однако они имеют тенденцию к децентрализации работы: множество пользователей, множество хранилищ, большие объемы параллельной разработки. Нам кажется, что традиционная модель с центральным хранилищем лучше всего подходит для управления версиями в контексте системного администрирования.

Можно воспользоваться также и некоторыми коммерческими системами управления изменениями. С какой-нибудь одной из них вы, наверное, уже имели дело, если вам приходилось работать в отделе разработки, и вы пытались адаптировать ее к вашим административным данным. Однако с ними нужно вести себя очень осторожно; наш опыт подсказывает, что коммерческие системы обычно крайне сложны для использования в системном администрировании.

Если вы еще не знакомы ни с одной из этих систем, мы рекомендуем вам начать с RCS, чтобы освоиться с управлением изменениями. Если вам нужно создать центральное хранилище для информации, используемой в системном администрировании, то лучше выбрать систему Subversion.

RCS: система контроля изменений

RCS — одно из старейших приложений UNIX. Эта довольно простая система работает с отдельными файлами и фиксирует изменения каждого контролируемого объекта в специальном скрытом файле. Имя этого файла совпадает с оригиналом, только к нему добавляется обозначение `,v`. Например, если файл `/etc/syslog.conf` контролируется системой RCS, то информация о его изменениях будет сохраняться в файле `/etc/syslog.conf,v`.

Чтобы не возникал беспорядок, система RCS ищет подкаталог `RCS` в том же каталоге, где находится исходный файл. Если таковой существует, файл `,v` создается там. Это чрезвычайно удобный подход, который мы настоятельно рекомендуем. Он позволяет не засорять основной каталог, так как под контролем системы RCS может находиться множество файлов.

Основная концепция системы RCS заключается в том, что пользователь сначала "выписывает" (check out) файлы, которые предстоит модифицировать. Затем он "регистрирует" (check in) изменения для их последующего сохранения. Соответственно, единственные команды RCS, которые действительно нужно знать, — это `co`, выписывающая файл, `ci`, регистрирующая измененный файл, и `rcs`, выполняющая различные служебные действия. Если использовать редактор `emacs`, вы сможете вообще не прибегать к услугам утилит командной строки, так как этот редактор имеет внутреннюю поддержку системы RCS.

Чтобы перевести файл под контроль системы RCS, достаточно ввести команду `ci`, чтобы зарегистрировать его. Например:

```
% ci -u syslog.conf
RCS/syslog.conf,v <-- syslog.conf
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> This is the syslog configuration file.
```

Флаг -u заставляет команду ci немедленно перевести файл syslog.conf в нередактируемое состояние. Если его не указать, команда ci зарегистрирует файл и удалит его исходную копию, хотя этого не требуется.

Каждый раз, когда вы собираетесь изменить файл, контролируемый системой RCS, выписывайте и блокируйте его командой co -l:

```
% co -l syslog.conf
RCS/syslog.conf,v --> syslog.conf revision 1.2 (locked)
done
```

Эта операция сообщит системе о предстоящем редактировании файла. Система не позволит никому другому выписать файл, пока вы не зарегистрируете его обратно.

RCS запрещает доступ на запись незаблокированных файлов, чтобы их нельзя было редактировать до тех пор, пока они не будут выписаны. Часто пользователи допускают такую ошибку: загружают файл в редактор, производят изменения, но не знают, что перед тем как редактор сможет сохранить изменения, файл нужно выписать. Чтобы такого не случилось, нужно просто приостановить работу редактора или открыть другое окно оболочки, выполнить команду co -l, а затем возобновить операцию сохранения. С помощью команды chmod можно было бы сделать файл перезаписываемым, однако это привело бы к сбою в работе RCS.

Теоретически подобная блокировка не позволяет двум пользователям одновременно редактировать файл. Но на практике для модификации системных файлов нужно быть суперпользователем, поэтому любой, кто имеет право выполнять команду sudo, сможет модифицировать файл, выписанный суперпользователем. В то же время, когда второй администратор попытается ввести команду co -l, система RCS сообщит, что версия файла, доступная для записи, уже существует, и выдаст сообщение. Администраторы должны взять за правило выписывать RCS-файлы, которые они собираются модифицировать. Наличие файла, открытого для записи, означает: “Стоп! Кто-то другой уже выписал этот файл”.

Иногда выясняется, что кто-то отредактировал файл и оставил его заблокированным или, еще хуже, изменил файл в обход RCS. Чтобы узнать, какие изменения при этом были сделаны, воспользуйтесь командой rcsdiff, которая является аналогом diff для RCS:

```
% rcsdiff syslog.conf
===================================================================
RCS file: RCS/syslog.conf,v
retrieving revision 1.3
diff -r1.3 syslog.conf 4c4
< define(LOGHOST,moonbase)
---
> define(LOGHOST,spacelounge)
%
```

В качестве последнего средства снимите блокировку командой rcs -u *имя_файла*. Команда попросит вас пояснить мотивы такого поступка и отправит сообщение пользователю, который заблокировал файл (к сожалению, обычно это суперпользователь).

После внесения изменений зарегистрируйте файл обратно с помощью команды ci -u. Вам будет предложено ввести комментарий, в котором вы должны будете написать то, что вы только что сделали. Не пропускайте этот шаг, и не пишите в комментарии целые поэмы. Через пару лет, когда вы захотите выяснить, для чего было произведено определенное изменение, полезные комментарии облегчат вашу жизнь.

```
% ci -u syslog.conf
RCS/syslog.conf,v <-- syslog.conf
new revision: 1.3; previous revision: 1.2
enter log message, terminated with single '.' or end of file:
>> Started logging debug messages to track down problem
>> ^D
done
```

Просмотреть историю редактирования файла позволяет команда `rlog`:

```
$ rlog syslog.conf
RCS file: RCS/syslog.conf,v
Working file: syslog.conf
head: 1.3
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 3; selected revisions: 3
description:
----------------------------
revision 1.3
date: 2002/01/10 00:44:58; author: adam; state: Exp; lines: +1 -0
Started logging debug messages to track down problem
----------------------------
revision 1.2
date: 2000/07/19 08:23:10; author: evi; state: Exp; lines: +2 -0
Changed log destination to new logmaster
----------------------------
revision 1.1
date: 1998/03/14 11:13:00; author: matthew; state: Exp;
Initial revision
=============================================================
```

Если нужно узнать, как выглядел файл после второй операции редактирования, выпишите версию 1.2 с помощью команды `co -r`:

```
% co -r1.2 syslog.conf
RCS/syslog.conf,v --> syslog.conf
revision 1.2
done
```

Команда заменяет текущий файл `syslog.conf` его старой версией, поэтому не забудьте выполнить обычную команду `co` по окончании работы, иначе демон `syslogd` окажется в недоумении. Другой опцией может быть `co -rl.2 -p syslog.conf`, которая посылает содержимое запрошенной версии на стандартный вывод; к сожалению, проверить файл с другим именем невозможно. Никогда не выписывайте заблокированные копии старых версий (с помощью команды `co -l`), так как при этом создаются дополнительные ветви в дереве версий. Это бывает полезно при работе с исходными текстами, но не для системного администрирования. Чтобы не усложнять себе жизнь, просто игнорируйте разделы документации по RCS, в которых говорится о данной возможности.

Дополнительную информацию о системе RCS можно найти по адресу `http://www.cs.purdue.edu/homes/trinkle/RCS/`.

CVS: система параллельных версий

RCS имеет некоторые серьезные недостатки. Например, при редактировании файлов следует быть осторожным, чтобы не "наступить на пятки" вашему коллеге. Также нужно придерживаться довольно специфического набора процедур. И, наконец, модифицировать несколько файлов одновременно нельзя, поскольку система может столкнуться с незавершенными изменениями. Эти недостатки, а также некоторые другие, привели к тому, что была разработана новая система следующего поколения, называемая Concurrent Versions System (CVS, система параллельных версий), которая в последнее время используется в системах UNIX и Linux чаще всего.

Одна из главных идей, лежащих в основе CVS, заключается в том, что все проектные файлы (а также их исторические версии) хранятся в одном месте. Как и в системе RCS, сначала нужно выписать копии файлов, с которыми вы хотите работать, а по окончании работы их нужно зарегистрировать. Положительным свойством системы CVS является то, что в ней не осуществляются проверки на блокировки; несколько людей могут выписать и модифицировать одни и те же файлы в одно и то же время (отсюда и термин "параллельные" в названии системы).

Так как система CVS не запрещает нескольким пользователям модифицировать файл, она должна обеспечить способ объединения изменений во время регистрации файлов. Такое объединение обычно происходит автоматически, однако оно работает только для текстовых файлов, и только в том случае, если разные изменения являются совместимыми друг с другом. В противном случае CVS доверяет пользователю, совершившему регистрацию, разрешить конфликты вручную. Если несовместимые изменения производятся в текстовом файле, CVS вставляет полезные комментарии, которые показывают, где возник конфликт.

На этом давайте закончим разбираться с теорией. Все звучит вроде бы здорово, и это действительно так на самом деле — в этой системе даже слияние работает нормально. Тем не менее, CVS имеет и ряд недостатков.

- Она не поддерживает "атомарные фиксации". Если два человека одновременно пытаются зарегистрировать серьезное изменение, которое повлияет на множество файлов, то две версии этого изменения могут быть приняты наполовину, причем каждому оператору будет выдан случайный набор возникших конфликтов, требующих разрешения. Естественно, в этом нет ничего хорошего.
- Чтобы переименовать файл, вы должны скопировать его под новым именем и удалить исходный файл, теряя, таким образом, всю прежнюю хронологию о файле. Точно так же и с каталогами: их можно переименовать только посредством копирования и удаления исходной версии.
- Изменения атрибутов файлов не подлежат управлению. Они остаются неизменными и имеют те настройки, которые имел файл во время первой регистрации.

Тем не менее, CVS применяется во многих программных проектах с открытым исходным кодом. Система используется часто не потому, что она слишком привлекательна, а потому, что просто нет другого альтернативного варианта. Однако ситуация изменилась, когда многие группы разработчиков стали работать над заменой CVS. Сегодня разрабатывается несколько вариантов систем, и какая-то одна из них станет наследницей CVS. На Web-сайте *Better SCM Initiative* (`www.better-scm.berlios.de`) Шломи Фиша (Shlomi Fish) можно познакомиться с новыми разработками и почитать результаты систематического сравнения их возможностей.

Ниже следует краткий перечень команд CVS, являющихся важными для пользователя. Первым шагом в модифицировании проекта является регистрация на сервере и выписка модуля, с которым планируется работать. Здесь мы работает с модулем `sort`.

```
$ cvs -d :pserver:имя_пользователя@имя_сервера:/путь/к/хранилищу login
CVS password: <пароль>
$ cvs -d :pserver: имя_пользователя@имя_сервера:/путь/к/хранилищу co sort
```

Здесь `pserver` — это метод доступа, используемый для контактов с хранилищем, которым в данном случае является сервер паролей CVS. Операция `login` сверяет пароль с паролем на сервере и создает его копию для использования в последующих транзакциях. Операция `co` является прямой аналогией такой же операции в системе RCS.

Сейчас вы можете войти в вашу локальную копию каталога `sort` и приступить к редактированию файлов. Когда вы будете готовы к регистрации файлов в хранилище CVS, вам не нужно будет использовать переключатель `-d`, поскольку CVS создала копию всей необходимой информации о местонахождении в подкаталоге `sort/CVS`.

```
$ cd sort
$ vi foo.c
$ cvs commit foo.c -m "Added more efficient sort routine"
```

Если вы работали со собственной копией модуля, и хотите обновить свои локальные копии файлов, которые были изменены (и зарегистрированы) другими пользователями с момента их выписки, можно использовать команду `cvs update`. Опция `-d` означает, что будут включены все подкаталоги, а опция `-P` дает указание системе CVS удалять любые пустые каталоги:

```
$ cvs update -dP
```

Имейте в виду, что хотя вы не регистрировали ваши файлы в центральном хранилище, эта операция является разновидностью объединения. Всегда существует вероятность того, что другие пользователи произведут изменения, которые будут конфликтовать с вашими изменениями. Система CVS даст вам знать о существовании конфликтов, которые нужно разрешить.

Subversion: отдаем должное CVS

Несмотря на то что CVS в настоящее время является доминирующей системой управления изменениями, мы полагаем, что администраторы в конечном счете выберут систему Subversion. Этот пакет был написан Карлом Фогелем (Karl Fogel) и Джимом Бланди (Jim Blandy), которые с 1995 г. продавали контракты на обслуживание системы CVS в фирме Cyclic Software. В 2000 г. они подписали контракт с фирмой CollabNet на написание системы с открытым исходным кодом, которая могла бы заменить CVS. В результате появилась система Subversion, а в феврале 2004 г. вышла ее первая версия 1.0.

В системе Subversion были учтены все упомянутые выше недостатки CVS, и это никак не сказалось на простоте ее использования. Как и в системе CVS, в центральном хранилище хранятся все файлы, управляемые версиями. Система Subversion может обрабатывать бинарные файлы и работает быстрее, чем CVS.

По умолчанию сервер Subversion является модулем в Web-сервере Apache 2.x. Это удобно для распределенной разработки программ, но, не удобно для администрирования. К счастью, почитатели Subversion предлагают другой тип сервера в виде демона `svnserve`. Этот демон можно запускать из домашнего каталога, экспериментируя с

системой Subversion, однако в производственных целях он должен иметь свою учетную запись пользователя и должен запускаться из `inetd`.

В первом выпуске Subversion использовалась база данных Berkley DB в качестве резервного места для ее хранилища, а Subversion 1.1 поддерживает альтернативную систему, известную как FSFS. В каждом случае есть свои преимущества и недостатки. Одним из главных недостатков является то, что Berkley DB основана на семантике ввода-вывода с отображением в памяти, поэтому в данном случае невозможно использовать NFS[7]. Хранилища Subversion, использующие Berkley DB, должны быть локальными по отношению к компьютеру, на котором работает сервер Subversion. Хранилища FSFS не имеют такого ограничения. О других достоинствах и недостатках можно почитать на сайте `subversion.tigris.org`.

Настроить хранилище несложно. Например, ниже показан пример создания нового хранилища Subversion, называемого `admin`.

```
# cd /home/svn
# mkdir repositories
# cd repositories
# svnadmin create admin
# chmod 700 admin
```

Если для вашего хранилища вы хотите использовать формат FFS вместо стандартного Berkley DB, добавьте опцию `--fs-type=fsfs` в команду `svnadmin create`. Однако следует иметь в виду, что в случае неверного выбора формат хранилища изменить будет сложно, поэтому выбор следует делать обдуманно.

Если заглянуть в каталог `admin`, то в нем вы сможете найти хорошо организованную структуру хранилища, включая файл `README`. Конфигурационный файл `svnserve.conf` можно найти в подкаталоге `conf`. Этот файл сообщает демону сервера о том, как нужно предоставлять доступ к новому каталогу. Ниже показан пример конфигурации, подходящей для административных файлов:

```
[general]
anon-access = none
auth-access = write
password-db = passwd
realm = The Sysadmin Repository
```

Так как одной из целей, стоящих перед разработчиками Subversion, было создание условий для совместной работы пользователей из разных организаций, в этой системе есть модель управления доступом, которая отделена от модели в операционной системе. Файл `passwd` (в том же каталоге) содержит список пользователей и их пароли (в открытом тексте!). Использование открытого текста нельзя отнести к достоинствам; с другой стороны, смягчающим обстоятельством является то, что пароли никогда не передаются по сети. Кроме того, пароли никогда не вводятся пользователями по памяти, поэтому можно присваивать многосимвольные пароли, длина которых выбирается случайным образом. Как следствие, такие пароли должны быть безопасными. Взгляните на следующий пример:

```
[users]
tobi = lkadslfkjasdljkhe8938uhau7623rhkdfndf
evi = 09uqalkhlkasdgfprghkjhsdfjj83yyouhfuhe
fritz = kd939hjahkjaj3hkuyasdfaadfk3ijdkjhf
```

[7] Более подробно об этом см. на сайте `www.sleepcat.com`.

Естественно, разрешения на файл `passwd` нужно устанавливать отдельно.

Все что осталось — это запустить сервер в отдельном хранилище:

```
# svnserve --daemon --root /home/svn/repositories
```

Будучи непривилегированным пользователем, вы можете сейчас выписать архив `admin` из любого места в сети:

```
$ svn checkout --username tobi svn://server.atrust.com/admin checkout
Authentication realm: <svn://server.atrust.com:3690> The Sysadmin Repository
Password for 'tobi': <пароль>
```

Когда вы введете пароль первый раз, система Subversion создает его копию в каталоге `.subversion`, который она создает в вашем домашнем каталоге. Чтобы добавить или удалить файлы из локальной копии проекта, используйте команду `svn`:

```
$ cd checkout
$ vi foo.c
$ svn add foo.c
```

Когда все будет готово, нужно зафиксировать изменения в хранилище:

```
$ svn commit -m "Initial checkin; added foo.c"
```

Перечислять измененные файлы, которые вы хотите зафиксировать, не нужно, хотя при желании это можно сделать; `svn` разберется с этим сама. Если опустить опцию `-m`, `svn` запустит редактор, в котором вы сможете отредактировать сообщение о фиксации.

Чтобы получить самые последние обновления из хранилища, запустите `svn update` внутри проекта. Как и в системе CVS, Subversion выполняет операцию по объединению любых файлов, которые были изменены как в вашей локальной копии проекта, так и в главном хранилище. Файлы с неразрешимыми конфликтами маркируются как “conflicted”, и Subversion не разрешит регистрировать их до тех пор, пока вы не устраните проблемы и не сообщите системе, что конфликты были разрешены:

```
$ svn resolved foo.c
```

Если вы хотите узнать, кто производил изменения, и какие строки в файле были изменены, спросите об этом систему Subversion:

```
$ svn blame bar.c
```

Эта команда печатает снабженную ссылками версию файла, которая показывает, когда и кем была изменена строка последний раз. (Оптимисты, а также те, кто терпит чужие ошибки, могут использовать синоним `svn pralse`.) Кроме того, можно без труда узнать о различиях относительно некоторой даты или версии. Например, если вы хотите узнать, что было изменено в файле `foo.c`, начиная с 2 июня 2006 г., вам поможет следующая команда:

```
$ svn diff -r "{2006-06-02}" foo.c
```

Последнюю версию системы Subversion можно загрузить на сайте `subversion.tigris.org`. Стандартной документацией является книга *Version Control with Subversion*, опубликованная в издательстве O’Reilly. Полный текст этой книги доступен в интерактивном режиме на сайте `svnbook.red-bean.com`.

Subversion имеет также знаменитый интерфейс Windows (зайдите на сайт `tortoisesvn.tigris.org`). Мы использовали его для управления исходными файлами для этой книги.

11.6. Локализация и конфигурирование

Адаптация компьютеров к вашей локальной среде является одним из основных заданий системного администрирования: вы должны сообщить системе обо всех принтерах, доступных в сети, запустить специальный демон лицензирования, добавить задачу `cron`, очищающую каталог `/scratch` один раз в неделю, интегрировать поддержку для специального сканера и т.п. Для выполнения этих задач нужен структурированный и воспроизводимый способ.

Вам нужно помнить о следующем.

- У пользователей нет привилегий системного администратора. Необходимость в них в ходе выполнения обычных операций вызывает подозрение и наверняка свидетельствует о том, что кто-то имеет злой умысел.
- У пользователей нет намерений причинить вред системе. Вы должны организовать защиту системы таким образом, чтобы она защищала ее от неумышленных ошибок и не передавала "первому встречному" привилегии системного администратора.
- Прежде чем наказывать пользователей, изредка делающих что-то не так, нужно провести с ними беседу. На неумелые действия администратора пользователи зачастую реагируют неадекватно. Следовательно, если ваши действия мешают пользователям работать, то это свидетельствует о проблемах в архитектуре.
- Старайтесь ориентироваться на заказчика. Беседуйте с пользователями и постарайтесь узнать, какие задачи для них являются сложными. Попытайтесь сделать эти задачи гораздо проще.
- Ваши предпочтения — только ваши, и ничьи более. Пользователи должны иметь собственные предпочтения. По мере возможности такие варианты нужно предлагать всегда и везде.
- Если ваши решения, принимаемые в административных целях, влияют на способность пользователей обучаться системе, постарайтесь их изменить. Расскажите о них пользователям.
- Постоянно обновляйте вашу локальную документацию и делайте ее легкодоступной. Более подробно об этом можно прочитать в разделе 30.8.

Организация локализации

Когда в организации тысяча компьютеров с различными конфигурациями, администратор вынужден тратить большую часть времени на выяснение причин того, почему такая-то проблема возникла только на этом компьютере. Думаете, решением проблемы будет унификация всех конфигураций? Не спешите с выводами! Ограничения, существующие в реальном мире, а также различные нужды ваших пользователей делают это невозможным.

Между администрированием "многочисленных" конфигураций и "бесчисленных" конфигураций есть разница. Важно разделить вашу установку на управляемые части. Вы увидите, что одни части локализации применяются ко всем управляемым компьютерам, другие применяются только к некоторым из них, а третьи — только к отдельным компьютерам.

Помимо выполнения инсталляций с "чистого листа", вам придется также постоянно загружать и инсталлировать обновления. Следует иметь в виду, что на разных компьютерах установлены различные сроки действия и предъявляются разные требования к стабильности и периоду работоспособности.

Предусмотрительный системный администратор никогда не станет развертывать новые выпуски программного обеспечения на всех компьютерах сразу. Наоборот, это нужно делать с учетом нужд каждой группы и потратить некоторое время на выявление проблем на самых ранних этапах, чтобы не допустить серьезного повреждения системы. Важные серверы нельзя обновлять до тех пор, пока не будет уверенности в надежности производимых изменений[8].

Однако в процессе проектирования системы локализации следует убедиться в том, что все исходные данные хранятся в системе управления изменениями. Так вы сможете всегда знать, какие изменения были тщательно проверены и готовы к развертыванию. Кроме того, это позволит вам идентифицировать пользователя, являющегося автором проблематичных изменений. Чем больше человек вовлечено в этот процесс, тем более важным является последнее утверждение.

Свои преимущества есть и у разделения базового выпуска ОС и выпуска локализации. В зависимости от того, какую степень стабильности необходимо достичь в вашей среде, вы можете использовать второстепенные локальные выпуски только для того, чтобы устранить ошибки. Наш опыт подсказывает, что добавление новых функций небольшими порциями приводит к более стабильному выполнению операций, чем при постановке изменений в очередь на следующие крупные выпуски, что сразу же снизит качество обслуживания.

Часто имеет смысл определять максимальное количество "выпусков", с которыми вы хотите работать в любой данный момент времени. Некоторые администраторы придерживаются того мнения, что если программное обеспечение работает надежно, то ничего изменять в нем не нужно. Они полагают, что не имеющая под собой почвы модернизация систем стоит времени и денег, и что "передовой выпуск" слишком часто означает "выпуск с множеством ошибок". Те же, кто вооружается этими принципами на практике, должны быть готовы к тому, что активные выпуски придется коллекционировать во вместительном каталоге.

В отличие от них, активные пользователи ссылаются на присущую выпускам сложность и трудность в осмыслении (не говоря уже об управлении) случайной коллекции выпусков, вышедших много лет тому назад. Их козырем являются "заплаты" системы защиты, которые должны обычно применяться универсальным образом и в строгом порядке. Инсталлировать "заплаты" в устаревших версиях операционной системы часто бывает просто невозможно, поэтому администраторы сталкиваются перед выбором: не инсталлировать обновления на некоторые компьютеры или полностью переходить на новые внутренние выпуски. Это не сулит ничего хорошего.

Ни одна из этих перспектив не является доказуемо верной, хотя мы стараемся придерживаться того мнения, что лучше использовать ограниченное количество выпусков. Лучше выполнять модернизацию по вашему собственному плану, чем руководствоваться чьим-то планом.

[8] Вероятным исключением из этого правила являются "заплаты" системы защиты. Нужно стараться закрывать все бреши в защите сразу же после их выявления. С другой стороны, "заплаты" системы защиты могут сами вызвать ошибки в работе системы.

Тестирование

Важно протестировать изменения, пока их еще можно отменить. По меньшей мере, это означает тестирование собственных настроек. Но в той же степени это касается и системного программного обеспечения. Один известный поставщик UNIX однажды выпустил "заплату", которая, будучи примененной определенным образом, выполняла команду `rm -rf /`. Представьте последствия установки такой "заплаты" во всей организации без предварительного тестирования.

Тестирование особенно полезно, когда применяются такие средства, как `apt-get` или Red Hat Network, способные автоматически устанавливать "заплаты". Не стоит подключать компьютеры, ответственные за выполнение ключевых задач, к службе обновления, спонсируемой поставщиком. Постарайтесь подключить к этой службе простой компьютер и уже с него распространяйте изменения на остальные компьютеры в вашей организации. Не допускайте производство обновлений во время проверки, иначе они могут раньше времени проникнуть в вашу производственную систему.

📖 О выявлении проблем речь пойдет в разделе 30.9.

Если вы предвидите возникновение каких-либо проблем, с которыми могут столкнуться пользователи, или знаете, что скоро будут внесены изменения, связанные с запланированным обновлением, заранее уведомьте об этом пользователей и позвольте им связаться с вами, если у них возникнут какие-либо трудности. Позаботьтесь о том, чтобы пользователи имели возможность оперативно сообщать о замеченных ошибках.

В транснациональной компании можно подключить к тестированию иностранные офисы. Участие пользователей из других стран особенно важно в случае многоязыковых систем. Если никто в американском офисе не говорит по-японски, необходимо связаться с офисом в Токио и попросить их проверить все, что касается поддержки раскладки кандзи (kanji). Вообще, на удивление большое число системных параметров зависит от региональных особенностей. К примеру, при тестировании систем печати имеет смысл проверять, что произойдет, если задать иной размер бумаги. Ведь в разных странах может использоваться офисная бумага разного формата.

Локальная компиляция

Давным-давно, когда существовало множество разных аппаратных платформ UNIX, программы обычно распространялись в виде архивов исходных текстов. Как правило, это были файлы с расширением `.tar.Z`, которые нужно было разархивировать с помощью команды `uncompress`, а потом скомпилировать. Затем программу требовалось инсталлировать в соответствующий каталог, например `/usr/local`. Сегодня, в эпоху систем управления пакетами, все меньше программ инсталлируется подобным образом. Задачи администраторов тоже сильно упростились, так как в пакетах указано, куда их инсталлировать.

Несмотря на простоту управления пакетами, многие предпочитают самостоятельно компилировать программы[9]. Это позволяет лучше контролировать параметры компиляции. Фанатики могут даже изучить исходный код, чтобы быть уверенными в его безопасности. Кому-то это кажется важным, но мы считаем, что если у вас нет времени и

[9] Для этой цели можно воспользоваться дистрибутивом Gentoo Linux, который можно перекомпилировать с "чистого листа".

достаточного опыта для анализа каждой из 20 000 строк приложения, то такая мера предосторожности приносит минимальную пользу.

Далеко не все программы выпускаются в виде пакетов для всех дистрибутивов Linux, поэтому вам наверняка придется столкнуться с программами, которые необходимо инсталлировать вручную, особенно если речь идет о 32-разрядных компьютерах на платформе Intel. В организации, разрабатывающий собственное программное обеспечение, необходимо также решить, куда его инсталлировать.

Исторически сложилось так, что локальные программы чаще всего помещают в каталог `/usr/local`, и это соглашение до сих пор широко распространено. Применяемый в UNIX/Linux стандарт FHS (Filesystem Hierarchy Standard — стандарт иерархии файловой системы) определяет, что после начальной инсталляции операционной системы каталог `/usr/local` должен присутствовать и быть пустым.

Многие пакеты "ожидают", что их установят именно туда. К сожалению, удручающе большое число пакетов (особенно коммерческих приложений) не следует данному правилу, "предпочитая" быть установленными в каталог `/usr`. Мы считаем это плохой идеей и не советуем так поступать, если это выходит за рамки контекста систем управления пакетами[10]. Наш опыт подсказывает, что хотя многие приложения стараются инсталлировать в каталог `/usr`, они работают без проблем, если инсталлированы где-то в другом месте. В редких случаях, когда пакет ведет себя ошибочно, можно устранить проблему, добавив в каталог `/usr` символические ссылки, которые будут указывать на другой каталог для инсталляции. Можно поступить иначе: нужно проверить, ссылается ли проблемное приложение на переменную окружения, и отыскать ее инсталляционный каталог; такая особенность обычно документируется, но не всегда.

Каталог `/usr/local` хоть и традиционен, но управлять им не всегда легко. Его способ организации (исполняемые файлы помещаются в подкаталог `/usr/local/bin`, man-страницы — в подкаталог `/usr/local/man` и т.д.) таков, что возникает целый ряд проблем: трудно инсталлировать несколько версий одной и той же программы, подкаталоги сильно разрастаются, сложно поддерживать несколько платформ и т.д.

Распространение локализаций

Система локализации в вашей организации должна справляться как с исходной инсталляцией, так и с последующими обновлениями. Особую сложность могут представлять обновления. Основным моментом в этом вопросе является достижение эффективности, поскольку нет смысла повторять всю локализацию с самого начала, чтобы обновить разрешения для одного файла. Несмотря на то что этот процесс производится автоматически, модель с повторной сборкой с "чистого листа" делает обновление дорогим и длительным во времени процессом.

Простой и масштабируемый способ организации локализации заключается в поддержке файлов в структуре дерева, которая напоминает файловую систему производственного компьютера. Специальный инсталляционный сценарий может скопировать дерево на искомый компьютер и выполнить все необходимые дополнительные действия по редактированию.

[10] Смысл хранения случайных программ вне каталога `/usr` и других системных каталогов сводится к тому, чтобы отделить локальные настройки от системных программ. В системе управления пакетами сами пакеты ведут контрольные записи, поэтому для пакетов имеет смысл изменять системные каталоги.

Такой тип установки имеет несколько преимуществ. Вы можете поддерживать столько деревьев локализации, сколько это необходимо для реализации вашей локальной административной схемы. Некоторые из деревьев будут альтернативными, причем каждый компьютер получит только один доступный вариант. Остальные деревья будут использоваться в качестве оверлеев — их можно будет копировать поверх уже существующих деревьев. Деревья локализации могут перезаписывать файлы, если это будет необходимо, или могут полностью заменить их структуру. Каждое дерево, которое потенциально инсталлируется независимым образом, должно быть представлено в отдельном проекте управления изменениями.

Подход с использованием деревьев-оверлеев позволяет внести гибкость в реализацию. Если вы используете систему упаковки для распространения ваших локальных настроек, то оверлеи можно просто реализовывать в независимых пакетах. В пакет можно добавить соответствующие сценарии настройки и задать их таким образом, чтобы они запускались как часть процесса инсталляции.

Еще одна хорошая идея с реализацией заключается в использовании утилиты `rsync`, которая согласовывает искомые компьютеры с деревьями-оверлеями. `rsync` копирует только те файлы, которые являются устаревшими, поэтому она может быть эффективной для распространения инкрементных изменений. Такого поведения трудно добиться с помощью одной лишь системы упаковки. Утилита `rsync` описана в разделе 17.3.

Устранение проблем планирования

В некоторых организациях обновление всех систем производится в соответствии с обычным графиком. Несмотря на то что этот план может быть навязчивым, в нем есть одно преимущество: он может определить время, по достижении которого клиентские системы становятся устаревшими. В других организациях обновление систем производится во время их загрузки. Это относительно безопасный вариант, однако он может означать, что обновления будут производиться через продолжительные промежутки времени. В других организациях, где работают технически грамотные пользователи, пользователям разрешается самостоятельно обновлять свои системы. Такая схема тоже имеет много преимуществ, однако в ее случае нужно использовать еще одну схему в качестве вспомогательной, поскольку некоторые пользователи вообще никогда не будут обновлять свои системы.

Обновления можно извлекать из сервера обновлений (обычно посредством `cron`), или же их могут извлекать отдельные клиенты. Система извлечения дает клиентам больше возможностей для управления собственными расписаниями обновления. Например, организация, осуществляющая свои операции по всему миру, может перейти к обновлениям в ночное время с помощью системы извлечения; модернизация, производимая в полночь в США, соответствует модернизации, производимой в полдень в Азии.

В зависимости от того, сколькими компьютерами вы управляете, и насколько обширной является география вашей организации, вы можете настроить один или несколько (в этом случае речь идет об иерархии) серверов для распространения. Например, у вас может быть один главный сервер, с которого производится распространение среди подчиненных серверов, находящихся в каждом здании, а подчиненные серверы занимаются распространением среди клиентов. Для организаций, имеющих свои отделения в разных уголках земного шара, такая схема позволит существенно сократить объемы данных, передаваемых по глобальной сети.

11.7. Средства управления конфигурацией

Системы локализации обычно являются "доморощенными". Частично это объясняется тем, что все организации отличаются друг от друга, и в каждой из них есть свои особенности. Однако большой вклад вносит также синдром NIH ("not invented here" — придумано не здесь). Возможно, отсутствие господствующего инструмента с открытым исходным кодом для управления конфигурацией заставляет нас смотреть на эту проблему как на такую, которая выходит за рамки стандартизированных средств.

Тем не менее, такие средства существуют и стоят того, чтобы вы обратили на них внимание. В следующих разделах будут рассмотрены наиболее популярные системы в порядке их популярности и схожести.

`cfengine`: компьютерная иммунная система

Одной из наиболее известных утилит локализации является утилита `cfengine`, разработанная Марком Бургессом (Mark Burgess). Она была представлена как "компьютерная иммунная система", выполняющая свои операции на основе модели, в соответствии с которой выбирается способ конфигурирования системы. В случае обнаружения расхождений между моделью и реальностью, `cfengine` выполняет определенные действия, направленные на то, чтобы привести систему в соответствие. Благодаря этой модели, утилита `cfengine` действительно является полезной для постоянной поддержки конфигурации.

Утилита `cfengine` умеет создавать резервные копии файлов, которые она модифицирует, и может хранить специальный журнал своих изменений. Ее можно запускать в режиме "бездействие", в котором она описывает изменения, которые она может произвести, в действительности не реализуя их.

С помощью специального языка `cfengine` можно описать способ конфигурирования компьютеров. Это можно сделать, например, так: "Файл `xyz` должен существовать в `/etc`, иметь права доступа 644 и принадлежать супервользователю". Можно написать правила в зависимости от содержимого отдельных файлов. Например, можно определить, что `/etc/hosts` должен содержать строку "router 192.168.0.1". Утилита `cfengine` добавит эту строку, если она будет отсутствовать.

Язык конфигурации `cfengine` позволяет формулировать индивидуальные правила в зависимости от факторов, таких как имя компьютера, тип операционной системы или маска подсети. Эта особенность облегчает написание одного конфигурационного файла, который удовлетворит нужды всех компьютеров, находящихся в зоне вашей административной ответственности.

Ниже показан простой пример из мира UNIX. В нем проверяется, является ли `/bin` символической ссылкой на `/usr/bin` в компьютерах Sun, производится дополнительная проверка ссылок в старых "коробочных" версиях OSF, и удаляет из каталога `/var/scratch` любые файлы, возраст которых превышает семь дней:

```
control:
actionsequence = ( links tidy )
links:
sun4::
/bin -> /usr/bin
# other links
osf::
```

```
# some osf specific links
tidy:
/var/scratch pattern=* age=7 recurse=inf
```

На домашней странице утилиты `cfengine` (`www.iu.hio.no/cfengine`) можно найти описание этой утилиты.

LCFG: крупномасштабная система конфигурирования

LCFG первоначально была разработана Полом Андерсоном (Paul Anderson) в университете Эдинбурга (Edinburgh University) в 1993 г. Последний вариант этой системы известен под именем LCFG(ng); она стала популярна среди многочисленных пользователей за пределами этого университета. LCFG первоначально предназначалась для управления большими установками Red Hat, однако это не все, на что она способна — она может управлять также системами Mac OS X и Solaris. Web-сайт этой системы находится по адресу `www.lcfg.org`.

Как и `cfengine`, LCFG имеет специализированный язык конфигураций. Конфигурации всех управляемых компьютеров хранятся на центральном сервере в виде подборки главных конфигурационных файлов. На основе этих файлов LCFG генерирует специализированные XML-файлы, описывающие конфигурацию каждого управляемого компьютера. Демон центрального сервера наблюдает за изменениями, происходящими в главных конфигурационных файлах, и генерирует необходимые XML-файлы.

XML-файлы публикуются на внутреннем Web-сервере, из которого клиенты могут извлекать свои собственные конфигурации. Клиенты используют разнообразные сценарии компонентов для своего конфигурирования в соответствии с XML-образцами.

Arusha Project (ARK)

Система Arusha Project была разработана Уиллом Параином (Will Parain) в университете Глазго (Glasgow University). Мотивация, лежащая в основе создания этой системы, заключалась в следующем: большинство администраторов делают по сути одно и то же, но не все. Или, как говорил автор системы, “я хочу, чтобы моя служба была такой же, как и у остальных администраторов, и имела лишь небольшие отличия”. Web-сайт этой системы находится по адресу `www.ark.sf.net`. Посетив страницы этого сайта, вы сможете узнать о философии этой системы.

Система Arusha Project основана на объектно-ориентированном языке, называемом ARK. Он выражается в виде XML и позволяет администратору определять такие объекты, как узлы, лицензии, Web-сайты, списки рассылок и т.п. С помощью ARK можно описывать зависимости, существующие между этими объектами, и создавать методы и данные, проясняющие их взаимоотношения. Как и в разработке программ, ведущейся с использованием объектно-ориентированных языков, можно повторно использовать объекты подклассов, заменяя только те части, которые необходимо изменить для нормального функционирования в вашей локальной среде.

Template Tree 2: помощник `cfengine`

Система Template Tree 2 была создана в Швейцарском федеральном технологическом институте (Swiss Federal Institute of Technology) Тобиасом Оэтикером (Tobias Oetiker). Она построенная на основе компонентов и управляется центральной конфигурацией. Ее изюминка заключается в том, что она может использовать двухуровневый подход в

определении конфигурации организации и может работать с перемещенными корневыми каталогами на бездисковых компьютерах.

На нижнем уровне система состоит из множества "пакетов с функциональными особенностями". Такой пакет представляет собой коллекцию файлов, сопровождаемых файлом `META`, который описывает, как эти файлы должны быть инсталлированы в искомой системе. В качестве функциональной особенности может быть что угодно, начиная от сетевой конфигурации, и заканчивая последней версией OpenSSH. Функциональные особенности могут предлагать конфигурируемые параметры, которые можно задавать в главном конфигурационном файле.

На верхнем уровне конфигурации представлен главный конфигурационный файл организации, в котором функциональные особенности собраны вместе, и который связывает их с компьютерами или группами компьютеров. На этом уровне вы должны определить значения для свободных конфигурационных параметров, предлагаемых каждой особенностью. Например, одним из параметров для почтового сервера может быть имя почтового домена.

Template Tree 2 комбинирует информацию из главного конфигурационного файла и `META`-файлов отдельных функциональных особенностей для генерирования конфигурационного файла `cfengine` для всей организации. Так как каждая особенность должна содержать документацию о ее назначении и использовании, Template Tree 2 может также генерировать составную документацию.

DMTF/CIM: общая информационная модель

Группа Distributed Management Task Force (DMTF — группа распределенного управления), насчитывающая, по их собственному признанию, более 3000 активных членов, работает с 1992 г. над общей информационной моделью (Common Information Model, CIM), пытаясь разработать стандарты для объектно-ориентированной, межплатформенной системы управления.

Выражаясь словами DMTF, CIM — это "схема управления ... предназначенная для установления общей концептуальной структуры на уровне фундаментальной топологии по отношению как к классификации и ассоциации, так и к базовому набору классов, ориентированных на установление общей структуры для описания управляемой среды". Или что-то в этом роде.

Членами DMTF являются все основные производители, начиная от Microsoft, и заканчивая Sun. К сожалению, предложенные ими стандарты, показывают, что эти фирмы в совершенстве овладели искусством сбивать с толку и вводить модные словечки. Фирмам, принимающим участие в этом проекте, и имеющим солидную репутацию, не к лицу демонстрировать свою привычку стандартизировать все подряд, без разбору. Стандарты сосредоточены вокруг XML и объектной ориентации. Однако нам пока что приходится видеть основанный на них удобный продукт.

Поскольку выработкой стандартов занимаются всем известные производители коммерческих продуктов, то и результат является соответствующим. Так, например, Sun продает систему для управления бездисковыми клиентами на основе этих технологий вместе с операционной системой Solaris 8. При ближайшем рассмотрении можно было увидеть сервер, основанный на Java, который запускался клиентом, написанным на Java, и запускал сценарий интерпретатора команд, который и выполнял всю работу. Если возникали проблемы, то в справочнике говорилось, чтобы пользователи обращались к регистрационному файлу, который представлял собой бинарную структуру, доступ к ко-

торой мог производиться только посредством приложения для просмотра журнала, написанном на Java.

Во всем этом безобразии есть один светлый момент: благодаря усилиям DMTF поставщики вынуждены предоставлять для своих систем, основанных на открытом стандарте, интерфейсы конфигурации, доступные программным образом. Для сред UNIX и Linux в этом нет ничего нового, однако и DMTF не является детищем UNIX. Членами DMTF являются Cisco, Microsoft, Symantec и многие другие компании, имеющие небогатый опыт в обеспечении удобных способов написания сценариев для своих систем. Если бы эти продукты получили API конфигурации, то это было бы здорово, даже если бы реализация была несовершенной.

11.8. Организация совместного использования программ через NFS

Где следует инсталлировать дополнительное программное обеспечение: на отдельных клиентах или на центральном файловом сервере, на котором его можно совместно использовать через NFS? Если речь идет о Linux, то ответ будет такой: "на клиентах". Однако NFS позволяет производить обновления гораздо быстрее (обновить десяток серверов NFS будет гораздо быстрее и надежнее, чем 1000 клиентов) и экономить место на дисках клиентов.

На самом деле вопрос сводится к сравнению управляемости и надежности. Доступ на основе сетевой файловой системы с каждым днем становится более централизованным и простым в управлении; с его помощью варианты устранения ошибок и новые пакеты становятся сразу же доступными на всех клиентах. Однако работа по сети может вестись медленнее, чем при доступе к локальному диску. Кроме того, в модели с сетевым сервером появляется зависимость от сети и файлового сервера, не только потому, что она добавляет потенциальные места сбоев, но и потому также, что она требует, чтобы клиенты и серверы согласовывали свои действия относительно таких вопросов, как доступ к общим библиотекам и версии этих библиотек. Недостатком является то, что эти библиотеки программ NFS представляют собой продвинутую технологию администрирования и должны быть опробованы только в тех средах, в которых координация действий производится из центра.

Если вы работаете в среде с эффективными сетями и можете создать быстрый RAID-массив для центрального сервера, вы обнаружите, что сетевой сервер в действительности работает быстрее, чем локальный диск IDE. Неизвестно, хорошо это или плохо, однако влияние на производительность является крайне низким; проблемы с производительностью не должны быть доминирующими для данного архитектурного решения.

Вообще, в сетях, состоящих из однородных систем, большая часть преимуществ достигается за счет совместно используемых хранилищ программного обеспечения. Если ваша организация стандартизировала операционную систему, и эта система имеет удобные функциональные средства для управления пакетами, то вам, наверное, лучше всего будет отказаться от использования родной системы.

Пространства имен пакетов

В традиционной системе UNIX содержимое новых пакетов распределяется по множеству каталогов. Библиотеки поступают в каталог `/usr/lib`, бинарные файлы — в ка-

талог /usr/bin, документация — в каталог /usr/share/docs и т.д. Linux наследует эти свойства в той или иной мере, хотя стандарт Filesystem Hierarchy Standard (Стандарт иерархии файловой системы) помогает сделать размещение предсказуемым. (Подробные сведения о FHS можно узнать на сайте www.pathname.com/fhs.)

Преимущество этого соглашения состоит в том, что файлы оказываются в хорошо знакомых местах. Например, если переменная окружения PATH указывает на каталог /usr/bin, и другие стандартные бинарные библиотеки, то новые программы будут доступны сразу же после инсталляции.

Недостатком является то, что источники файлов должны быть явным образом отслежены (средствами системы управления пакетами) и что разрозненные файлы трудно использовать совместно по сети. К счастью, системные администраторы могут выйти из затруднения благодаря пространствам имен пакетов.

Суть этой схемы заключается в том, чтобы инсталлировать каждый пакет в его собственный отдельный корневой каталог. Например, вы можете инсталлировать gimp в каталог /tools/graphics/gimp, а бинарные файлы — в каталог /tools/graphics/gimp/bin/gimp. Вы можете создать заново единый бинарный каталог для вашей коллекции утилит, поместив символические ссылки в каталог, такой как /tools/bin, например:

```
/tools/bin/gimp -> /tools/graphics/gimp/bin/gimp
```

Пользователи могут затем добавить каталог /tools/bin в свои переменные PATH, чтобы иметь возможность пользоваться всеми общими утилитами.

Определять структуру каталога /tools можно разными способами. Иерархический подход (например, /tools/graphics, /tools/editors и т.д.) облегчает обзор и повышает производительность. В соглашение об именовании можно включить версию программы, архитектуру оборудования, операционную систему или инициалы ответственных пользователей, чтобы одним набором утилит могли пользоваться многие типы клиентов. Например, пользователи Solaris могут включить в свои переменные PATH ссылку /tools/sun4/bin, а пользователи Fedora — ссылку /tools/fedora/bin.

Когда вы инсталлируете новую версию основной утилиты, то старые версии желательно хранить как можно дольше, особенно если пользователи подолгу работают в проектах, в которых используется данная утилита. В идеале, новые версии утилит должны быть совместимыми со старыми файлами данных и программным обеспечением, однако на практике иногда встречается обратное. Неплохо заставить пользователей решить некоторую проблему с конфигурацией, чтобы получить доступ к старой версии пакета; неприятно просто прервать их работу и вынудить иметь дело с последствиями.

Управление зависимостями

Некоторые пакеты зависят от библиотек или других программных пакетов. Когда вы инсталлируете программное обеспечение локальным образом посредством систем управления пакетами, то для решения этих проблем вы получите всестороннюю помощь. Однако когда вы будете создавать собственное сетевое хранилище программ для своей организации, вам придется решать эти проблемы самостоятельно.

Если вы управляете библиотеками таким же образом, как и приложениями, вы можете компилировать утилиты для использования библиотек из общего каталога /tools. Так вы сможете хранить активными одновременно множество версий одной библиотеки. Поскольку зависимые приложения связаны со специфическими версиями библиотеки, установка будет работать стабильно даже после выпуска новых версий библиотеки.

Недостатком является то, что этот тип установки может быть довольно сложным, чтобы его использовать и обслуживать в течение продолжительного периода времени.

Старайтесь не создавать ссылки на локальный каталог `/tools/lib`, содержащий ссылки с обобщенными именами на общие библиотеки. Если вы измените ссылки, у вас могут возникнуть неожиданные проблемы, трудно поддающиеся диагностике. Системы с совместно используемыми библиотеками призваны избавить администраторов от некоторых возможных трудностей, однако в сложных установках лучше не рисковать.

Действия, необходимые для того, чтобы компоновщик использовал конкретную версию совместно используемой библиотеки, могут отличаться от системы к системе. В Linux вы можете задать переменную окружения `LD_LIBRARY_PATH` или использовать опцию `-R` компоновщика.

Сценарии упаковщика

К сожалению, совместимость на уровне библиотек — это еще полдела. То обстоятельство, что утилиты вызывают одна другую напрямую, порождает другую возможность возникновения конфликтов. Например, предположим, что утилита `foo` часто использует утилиту `bar`. После обновления версии `bar`, используемой по умолчанию, может оказаться, что утилита `foo` неожиданно перестала работать. В этом случае вы можете прийти к выводу, что `foo` зависит от некоторого поведенческого аспекта `bar`, который уже не поддерживается (или, по крайней мере, не поддерживается по умолчанию).

Если ваше хранилище программного обеспечения поддерживает множество версий (например, `/tools/util/bar-1.0` и `/tools/util/bar-2.0`), вы можете устранить эту проблему, переместив исходную версию утилиты `foo` в `foo.real`, и заменив ее небольшим сценарием упаковщика:

```
#!/bin/sh
# сначала убедитесь в том, что программа находит любые файлы, упакованные
# вместе с ней, даже если она не использует явный путь.
PATH=/tools/util/bar-1.0/bin:$PATH
export PATH
exec /tools/util/foo-1.0/bin/foo.real "$@"
```

Теперь утилита `foo` будет запускаться с помощью специальной переменной окружения `PATH`, и она будет вызывать старую версию утилиты `bar`, а не новую.

Упаковщики представляют собой превосходные инструменты, которые могут справляться не только с зависимостями между пакетами, но и решать такие задачи, как защита, зависимость от архитектуры или операционной системы и отслеживание использования. В некоторых организациях упаковывают все совместно используемые библиотеки.

Средства реализации

Поддержка совместно используемых хранилищ программного обеспечения является обычной проблемой в системном администрировании, поэтому администраторы придумали разные программные решения, которые помогают в этом деле. Как и в случае с локализацией, предпочтение отдается “доморощенным” решениям; тем не менее, стоит разобраться с доступными системами, чтобы узнать, могут ли какие-либо из них удовлетворить ваши нужды.

Чтобы познакомиться с некоторыми классическими системами, поищите в Google системы CMU Depot, CERN ASIS или CITES Encap. Одни из них до сих пор живы-здоровы, а другие не выдержали конкуренции со стороны продуктов Windows и повсеме-

стной доступности обновлений, упакованных с помощью RPM. Относительно недавно появились следующие проекты.

- GNU Stow (`www.gnu.org/software/stow`) поддерживает ссылки из центрального двоичного каталога на настоящие библиотеки, находящиеся в каталогах, специфических для пакетов.
- Swpkg, разработанная Крисофом Кальтом (Chrisophe Kalt; `web.taranis.org/swpkg`). Это коллекция инструментов, позволяющих решить большинство проблем, которые возникают в процессе создания совместно используемого хранилища программ.
- Система Pack Management Project, разработанная Питером Кристенсеном (Peter Kristensen; `pack.sunsite.dk`) известна своей амбициозностью: она не только предлагала систему для инсталляции программ, но и саму систему предварительно упакованных программ (с прицелом на Solaris). От предварительной упаковки отказались сразу же, а с программными инструментами она работает, как и прежде.
- Система SEPP, созданная в Швейцарском федеральном технологическом институте (`www.sepp.ee.ethz.ch`), реализует большинство идей, упомянутых выше.

11.9. Рекомендуемые программы

Одни инструменты столь важны, что мы считаем их обязательными для всех компьютеров. Другие пакеты, которые нам нравятся, могут быть необходимыми или полезными только для выбранных компьютеров. С другой стороны, в наше время стоимость жестких дисков такова, что можно не скупиться и инсталлировать на них все подряд. В табл. 11.1 показан список того, что нужно иметь под рукой.

В табл. 11.2 представлены программы, которые нам нравятся, но которые не являются обязательными. Мы разделили таблицу на два раздела: для системных администраторов и для рядовых пользователей.

Таблица 11.1. Необходимые программные пакеты

Пакет	Описание и комментарии
`ssh/scp`	Secure shell, или безопасная оболочка — использует криптографию, не открывает пароли
`sudo`	Замена `su` — расширяет возможности управления и регистрации
`sendmail` или `postfix`	Программы для пересылки почтовых сообщений
`traceroute`	Трассировка маршрутов по сети — необходим для отладки проблем, возникающих в сети
`tcpdump` или `Wireshark`	Выбор для сетевых анализаторов пакетов — используется для анализа сетевого трафика
`gzip`	Компрессор данных GNU — необходим для распаковки загруженных данных
`mozilla` или `firefox`	Выбор для Web-браузеров
RCS/CVS/Subversion	Системы управления изменениями — полезен для системных администраторов и пользователей
Perl	Язык написания сценариев — решение обычных задач

Все программы, перечисленные в табл. 11.1 и 11.2, являются бесплатными. Многие из них доступны в сети (попробуйте зайти на сайт `freshmeat.net`), и лишь немногие доступны через FTP. Если их нет в вашем дистрибутиве, воспользуйтесь поисковым сервером, чтобы найти их, или полистайте внимательно страницы этой книги (загляните в индекс — он вам обязательно поможет).

Таблица 11.2. Полезные пакеты программ

	Пакет	Описание и комментарии
Для администраторов	`gcc`	Компилятор C/C++ — высококлассный компилятор из GNU
	BIND	Инструменты службы имен — получают текущие версии (необходимо для защиты)[а]
	`npasswd`	Замена `passwd` — помогает пользователям выбирать надежные пароли
	`xntpd`	Демон времени — следит за часами компьютера и синхронизирует их
	Samba	Windows SMB — совместно использует файлы и принтеры с системами Windows
	Apache	Web-сервер
	Squid	Web-прокси и программы для кэширования
	`imapd/procmail`	Почтовые утилиты — доступ и фильтрация почтовых сообщений
	`mrtg`, RRDtool	Средства мониторинга — для сетевого трафика и других данных
Для пользователей	Acrobat Reader	Отображение файлов в формате PDF, удобная программа от Adobe[б]
	`gimp`	Редактор битовых изображений
	`xfig`	Простая программа для рисования в системе X Window
	GnuPG	Подписывает, проверяет и зашифровывает сообщения
	`nvi/vim`	Текстовые редакторы, подобные `vi` — рекомендуются для системных администраторов
	`emacs`	Текстовый редактор/операционная система — подходит для продвинутых пользователей
	`pico`, `nedit`	Текстовые редакторы — отличный выбор для начинающих
	`enscript`, `mpage`	Утилиты для печати
	`pine`	Читает почтовые сообщения; подходит для начинающих
	`thunderbird`	Читает почтовые сообщения; подходит для тех, кто предпочитает графический интерфейс пользователя
	`mh/exmh`	Читает почтовые сообщения (может справиться с большим объемом сообщений; поддерживается недостаточно хорошо)
	`glimpse`	Инструмент индексирования — индексирует файлы и выполняет быстрый поиск
	`gs`, `gv`, `ghostview`	Средства для предварительного просмотра и печати документов PostScript

[а] Пакет BIND включает `dig`, `host` и `nslookup`.

[б] Бесплатный пакет Envice является подходящей заменой Acrobat Reader для GNOME.

11.10. Рекомендуемая литература

- Intel Corporation and SystemSoft. *Preboot Execution Environment (PXE) Specification, Version 2.1.* 1999. `www.pix.net/software/pxeboot/archive/pxespec.pdf`
- Mintha, Jim, AND Pieter Krul. *UltraLinux FAQ*. `www.ultralinux.org/faq.html`
- *PXELinux Questions*. `syslinux.zytor.com/pxe.php`
- Rodin, Josip. *Debian New Maintainers' Guide*. `www.debian.org/doc/maint-guide` Этот документ содержит хорошую информацию о пакетах формата `.deb`.
- Silva, Gustavo Noronha. *APT HOWTO*. `www.debian.org/doc/manuals/apt-howto`
- Hohndel, Dirk, and Fabian Hershell. *Automated Installation of Linux Systems Using YaST*. `www.usenix.org/events/lisa99/full_papers/hohndel/hohndel_html`
- Stückelberg, Marc Vuileumier, and David Clerc. *Linux Remote-Boot mini-HOWTO: Configuring Remote-Boot Workstations with Linux, DOS, Windows 95/98 and Windows NT*. 1999. `tldp.org/HOWTO/Remote-Boot.html`
- Wachsmann, Alf. *How to Install Red Hat Linux via PXE and Kickstart*. `www.stanford.edu/~alfw/PXE-Kickstart/PXE-Kickstart.html`
- *The Red Hat Enterprise Linux System Administration Guide*. `www.redhat.com/docs`
- Burgess, Mark. *"Cfengine: A Site Configuration Engine."* USENIX Computing Systems, Vol 8, No 3. 1995. `www.cfengine.org`
- Oetiker, Tobias. SEPP: *Software Installation and Sharing System*. Boston: LISA 1998. `people.ee.ethz.ch/oetiker/sepp`

11.11. Упражнения

11.1. Опишите различия между программами Kickstart и AutoYaST. В каких ситуациях предпочтительнее пользоваться альтернативными программами, а не встроенными инсталляторами дистрибутивов?

11.2. Инсталлируйте копию Subversion из сайта `subversion.tigris.org`. Установите `snvserve` и создайте хранилище. Как можно открыть хранилище, чтобы оно использовалось из любого места в сети, делая его при этом защищенным?

11.3. Выясните, как организовано хранение локального программного обеспечения в вашей организации. Легко ли масштабируется такая система? Удобно ли с ней работать? Аргументируйте свои ответы

☆ 11.4. Какие действия необходимо предпринять для создания пакета RPM? Проиллюстрируйте это на примере какой-нибудь своей программы.

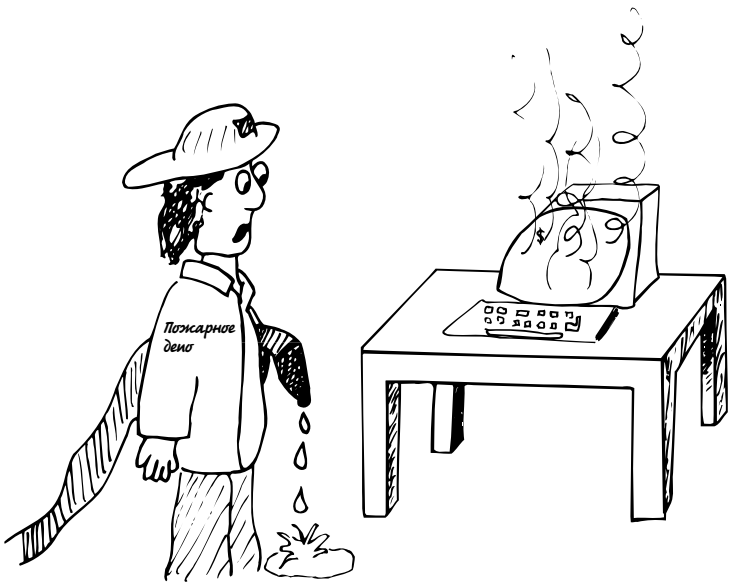
☆ 11.5. Повторите упражнение 11.4, но на этот раз создайте пакет в формате `.deb` для Debian.

☆☆ 11.6. Сконфигурируйте какой-нибудь сетевой инсталлятор по своему усмотрению и установите новую систему с локального сервера. Перечислите действия, которые пришлось предпринять для этого. Какие возникли трудности? Какие слабые места с точки зрения масштабируемости были вами замечены при работе с выбранным инсталлятором?

Book_LinuxAdmHand.indb 322 17.04.2007 13:55:10

ЧАСТЬ II

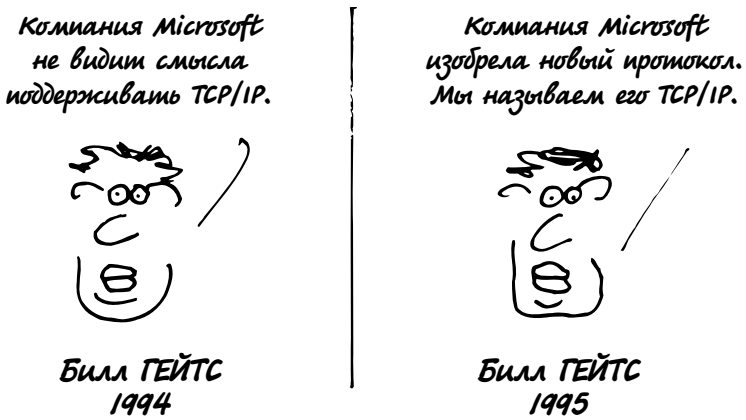
Работа в сетях



Book_LinuxAdmHand.indb 324 17.04.2007 13:55:10

Глава 12

Сети TCP/IP



Трудно переоценить важность сетей в современном компьютерном мире. Во многих организациях работа в Internet и доступ к электронной почте имеют огромное значение. По оценкам, к началу 2006 г. в Internet работало более одного миллиарда пользователей, что составило 55% прироста по сравнению с 2001 г. Повседневной задачей большинства системных администраторов является обслуживание локальных сетей, Internet-подключений, Web-сайтов и сетевого программного обеспечения.

TCP/IP — семейство сетевых протоколов, широко используемых в Linux/UNIX, MacOS, Windows и большинстве других операционных систем. Протоколы этого семейства составляют "язык" сети Internet. Ключевыми являются протоколы IP (Internet Protocol — межсетевой протокол) и TCP (Transmission Control Protocol — протокол управления передачей).

Благодаря TCP/IP формируется единый программный интерфейс доступа к разнообразному сетевому оборудованию, что позволяет обмениваться данными несмотря на различия в аппаратном и программном обеспечении. В основе сети Internet лежит протокол IP, используемый для доставки неструктурированных пакетов. TCP и UDP (User Datagram Protocol — протокол передачи дейтаграмм пользователя) являются "транспортными" протоколами. Они реализованы поверх IP и служат для доставки пакетов нужным приложениям.

Протокол TCP ориентирован на установление соединений. Он существенно упрощает ведение "диалога" между двумя программами. TCP-соединение напоминает телефонный разговор: информация, выдаваемая на одном конце, принимается на другом, и наоборот. Связь поддерживается даже в том случае, когда оба абонента молчат. Протокол

TCP обеспечивает гарантированную доставку пакетов, а также обладает средствами контроля трафика и управления перегрузкой.

Протокол UDP ориентирован на доставку одиночных пакетов без установления соединения. Этот процесс напоминает отправку письма. Поскольку протокол не предусматривает установления соединений, он не гарантирует, что пакеты будут получены в том порядке, в каком они отправлялись. Средства управления перегрузкой также отсутствуют.

TCP — "вежливый" протокол, заставляющий одновременно работающих пользователей делить друг с другом сетевой канал, что способствует повышению производительности. В противоположность этому протокол UDP старается передавать пакеты с максимально возможной скоростью.

По мере роста сети Internet все большую долю трафика должны занимать TCP-потоки, что позволит избегать заторов и эффективно использовать полосу пропускания. В настоящее время TCP-потоки составляют подавляющую часть трафика, а UDP- и ICMP-потоки соответственно занимают второе и третье место. Конечно, такие UDP-программы, как игровые, аудио- и видеоприложения по-прежнему широко используются в сети, но их трафик "перекрывается" трафиком Web-браузеров и программ вроде BitTorrent, которые работают по протоколу TCP, а не UDP, и занимают большую часть полосы пропускания.

В этой главе мы начнем знакомство с техническими и административными аспектами протоколов TCP/IP. К сожалению, даже изложение основ работы в сети — слишком обширная тема, чтобы ее можно было "втиснуть" в рамки одной главы. Поэтому мы продолжим эту тему в главах 13, 15, 19 и 20.

Несколько следующих разделов содержат основные теоретические сведения о протоколах и принципах построения Internet и поэтому носят достаточно общий характер. Подробное описание протокола IP начинается с раздела 12.4, а особенности конфигурирования сетевых параметров различных дистрибутивов приведены в разделе 12.8.

12.1. TCP/IP и Internet

История TCP/IP тесно связана с историей Internet и уходит корнями на несколько десятилетий назад. Популярность Internet во многом обусловлена гибкостью и эффективностью семейства TCP/IP, а также тем фактом, что это открытое и некоммерческое семейство протоколов. В свою очередь, широкое распространение TCP/IP именно в сети Internet позволило этому семейству одержать верх над несколькими конкурирующими семействами, популярными в свое время по тем или иным причинам.

Краткий исторический экскурс

Вопреки распространенному заблуждению, Internet — это не продукт компании Microsoft. С таким же успехом честь его создания можно было бы приписать первому вице-президенту США. Прародительницей Internet была сеть ARPANET, организованная в 1969 г. по инициативе DARPA (Defense Advanced Research Project Agency — управление перспективного планирования научно-исследовательских работ при Министерстве обороны США). Впоследствии ARPANET стала национальной магистралью сети NSFNET (National Science Foundation Network — сеть Национального научного фонда), которая соединяла между собой суперкомпьютеры и региональные сети США.

К концу 80-х гг. сеть перестала быть научно-исследовательской, поэтому Национальный научный фонд отошел от этого проекта. Переход на коммерческие рельсы занял несколько лет; сеть NSFNET прекратила функционирование в апреле 1994 г. Сегодня магистральная сеть образована совокупностью частных сетей, принадлежащих провайдерам Internet и соединяющихся в так называемых "точках обмена трафиком".

В середине 80-х гг. сеть Internet состояла из вычислительных центров ARPANET и группы университетских центров, в которых были установлены компьютеры DEC VAX с операционными системами семейства Berkeley UNIX. Университетские сети создавались по технологии Ethernet (скорость работы — 10 Мбит/с) и получали доступ в Internet по выделенным цифровым телефонным линиям со скоростью 56 Кбит/с. Каждый сентябрь, когда студенты приступали к учебе, глобальная сеть испытывала существенную перегрузку. Ван Якобсон (Van Jacobson), в то время научный сотрудник группы сетевых исследований при лаборатории Bell Labs университета Лоренса, заинтересовался поведением существующих протоколов в условиях повышенной нагрузки и нашел пути их оптимизации. Отсюда берут свое начало известные сегодня алгоритмы *медленного старта*, *предотвращения заторов*, *быстрой ретрансляции* и *быстрого восстановления*.

Повышение мощности вычислительных средств и рыночный спрос существенно ускорили развитие глобальной сети. С конца 80-х гг., когда были зафиксированы существующие на сегодняшний день алгоритмы протокола TCP, скорость сетевых интерфейсов возросла в 1000 раз (появление десятимегабитных технологий Ethernet привело к повышению эффективности на 6%, а десятигигабитных технологий — на 100%), выделенных линий — в 12 000 раз, а общее число хостов увеличилось в 80 000 раз.

Любой, кому случалось разработать программную систему, устаревшую сразу после появления следующего поколения аппаратных средств или новой версии операционной системы, не может не восхищаться тем, что сеть Internet до сих пор жива и успешно функционирует, эксплуатируя то же семейство протоколов TCP/IP, которое было разработано 30 лет назад для совершенно другой сети. Мы снимаем шляпы перед Бобом Каном (Bob Kahn), Винтом Церфом (Vint Cerf), Джоном Постелом (John Postel), Ваном Якобсоном и другими людьми, которые стояли у истоков этого чуда.

Кто сегодня управляет сетью Internet

Проектирование сети Internet всегда осуществлялось на кооперативных и открытых началах. Сегодня, когда глобальная сеть стала важным фактором мировой экономики, многие обеспокоены тем, что руководство сетью сосредоточено в руках группы людей, получающих указания от правительства США. Поэтому недалеко то время, когда появится правительство Internet.

В процесс управления глобальной сетью вовлечены следующие организации:

- ICANN (Internet Corporation for Assigned Names and Numbers — организация по назначению имен и адресов в Internet) — именно об этой организации можно сказать, что она отвечает за работу глобальной сети (`www.icann.org`);
- ISOC (Internet Society — сообщество пользователей Internet) — членская организация, которая представляет интересы пользователей Internet (`www.isoc.org`);
- IETF (Internet Engineering Task Force — проблемная группа проектирования сети Internet) — эта организация отвечает за разработку и публикацию технических стандартов Internet, являясь открытым форумом, принять участие в котором может любой желающий (`www.ietf.org`).

Среди перечисленных организаций наибольшая ответственность лежит на ICANN: она выступает в качестве руководящего органа Internet, исправляя ошибки, допущенные в прошлом, и продумывая пути дальнейшего развития глобальной сети.

Кроме перечисленных организаций существует также международный сетевой консорциум, получивший название Internet2, в который входят ученые-исследователи, правительственные организации различных стран и руководители промышленности. Эти организации координируют усилия по выработке концепций и разработке технологий, которые оказывают решающее влияние на развитие глобальной сети. Вопреки распространенному мнению, которое можно часто встретить в средствах информации, Internet2 — вовсе не является сетью, отдельной от Internet. Хотя в ее основе и лежит частная сеть Abilene, предназначенная для организации исследований с применением сетевых технологий, она не является заменой Internet. Подробные сведения о Internet2 можно найти на сайте `www.internet2.edu`.

Сетевые стандарты и документация

Научно-техническая деятельность сообщества пользователей Internet находит отражение в серии документов, известных как RFC (Request For Comments — запрос на получение комментариев). В формате документов RFC публикуются стандарты протоколов, предлагаемые нововведения, а также информационные бюллетени. Получая сначала статус черновых проектов (Internet Draft), после бурных дебатов в телеконференциях и на форумах IETF они либо отвергаются, либо получают официальный статус. Свое мнение по поводу предлагаемого стандарта может высказать любой участник телеконференции. Иногда документ RFC представляет собой всего лишь пояснение к существующим методикам и технологиям.

Документы RFC имеют порядковые номера. На момент написания этой книги их было более 4000. У каждого документа есть описательное название (например, *Algorithms for Synchronizing Network Clocks* — алгоритмы сетевой синхронизации часов), но во избежание неоднозначности на документы чаще всего ссылаются по номерам. Будучи опубликованным, документ RFC никогда не меняется. Изменения и дополнения публикуются в виде новых документов с собственными номерами. По существующему соглашению обновленные документы содержат всю информацию из предшествующих документов, которая остается актуальной. Таким образом, происходит полное замещение одного документа другим, по крайней мере, теоретически.

Сам процесс публикации документов RFC описан в документе под названием *Internet Official Protocol Standards* (официальные стандарты протоколов Internet). В него также включены ссылки на большинство текущих документов, содержащих описания протоколов. Поскольку эта информация часто меняется, документ повторно публикуется через каждые 100 номеров. Процесс публикации стандартов Internet описан в документе RFC2026. Другой полезный документ — RFC2555, *30 Years of RFCs* (30 лет существования документов RFC). В нем описаны социальные и технические аспекты публикации документов RFC.

Пусть читателей не пугает обилие технических подробностей в документах RFC. В большинстве из них содержатся вводные описания, резюме и толкования, которые будут полезны системным администраторам. Некоторые документы специально написаны в виде обзора или общего введения. Чтение документов RFC, возможно, не самый простой способ разобраться в той или иной теме, но это авторитетный, лаконичный и бесплатный источник информации.

Не все документы RFC написаны сухим техническим языком. Встречаются документы развлекательного содержания (часть из них опубликована первого апреля), среди которых нам особенно нравятся следующие:

- RFC1118 — *The Hitchhiker's Guide to the Internet* (руководство для путешествующих по Internet автостопом);
- RFC1149 — *A Standard for the Transmission of IP Datagrams on Avian Carriers* (стандарт передачи дейтаграмм птичьей почтой)[1];
- RFC1925 — *The Twelve Networking Truths* (12 сетевых истин);
- RFC2324 — *Hyper Text Coffee Pot Control Protocol (HTCPCP/1.0)* (гипертекстовый протокол управления кофеварками);
- RFC2795 — *The Infinite Monkey Protocol Suite (IMPS)* (семейство протоколов для управления бесконечным числом обезьян).

Это довольно забавные документы, которые приоткрывают характер людей, занимающихся проектированием и разработкой сети Internet.

Помимо собственных порядковых номеров документам RFC могут назначаться номера серий FYI (For Your Information — к вашему сведению), BCP (Best Current Practice — лучший существующий подход) и STD (Standard — стандарт). Перечисленные серии являются подмножествами серии RFC, группирующими документы по важности или назначению.

Документы FYI — это вводные или информационные материалы, предназначенные для широкой аудитории. Как правило, именно с них лучше всего начинать изучать незнакомую тему. Документы STD содержат описания протоколов Internet, прошедших процедуру проверки и тестирования в IETF и формально принятых в качестве стандартов. Документы BCP описывают рекомендуемые процедуры для администраторов Internet-серверов. Это наиболее ценные для системных администраторов документы в серии RFC.

Нумерация документов в рамках серий RFC, FYI, STD и BCP ведется раздельно, поэтому один документ может иметь несколько номеров. Например, документ RFC1635, *How to Use Anonymous FTP* (как создать анонимный FTP-сервер), известен также под номером FYI0024.

Документы RFC доступны по самым разным каналам. Список поддерживаемых зеркальных RFC-серверов можно получить на Web-сайте `www.rfc-editor.org`, который является центром сбора информации по документам RFC.

12.2. Семейство протоколов TCP/IP

Завершив краткий обзор, давайте подробнее рассмотрим, что собой представляют протоколы TCP/IP. TCP/IP — это семейство сетевых протоколов, ориентированных на совместную работу. В состав семейства входит несколько компонентов:

- IP (Internet Protocol — межсетевой протокол) — обеспечивает передачу пакетов данных с одного компьютера на другой (RFC791);

[1] Группа энтузиастов Linux из города Берген в Норвегии, называющая себя BLUG (Bergen Linux User Group), действительно реализовала протокол CPIP (Carrier Pigeon Internet Protocol — межсетевой протокол голубиной почты) в соответствии с документом RFC1149. Не верите — посетите их Web-сайт по адресу `www.blug.linux.no/rfc1149`!

- ICMP (Internet Control Message Protocol — протокол управляющих сообщений в сети Internet) — отвечает за различные виды низкоуровневой поддержки протокола IP, включая сообщения об ошибках, вспомогательные маршрутизирующие запросы и отладочные сообщения (RFC792);
- ARP (Address Resolution Protocol — протокол преобразования адресов) — обеспечивает трансляцию IP-адресов в аппаратные адреса (RFC823)[2];
- UDP (User Datagram Protocol — протокол передачи дейтаграмм пользователя) и TCP (Transmission Control Protocol — протокол управления передачей) — обеспечивают доставку данных конкретным приложениям на указанном компьютере. Протокол UDP реализует передачу отдельных сообщений без подтверждения доставки, тогда как TCP формирует надежный дуплексный канал связи между процессами на двух компьютерах с возможностью управления потоками и контроля ошибок (RFC768 и RFC793).

Семейство TCP/IP базируется на многоуровневой схеме взаимодействия протоколов (табл. 12.1).

Таблица 12.1. Сетевая модель TCP/IP

Уровень	Назначение
Прикладной	Пользовательские приложения
Транспортный	Доставка данных приложениям[a]
Сетевой	Базовые функции коммуникации, адресации и маршрутизации
Канальный	Сетевые аппаратные средства и драйверы устройств
Физический	Кабели и другие физические носители

[a] Дополнительно могут обеспечиваться подтверждение доставки и контроль передачи данных.

После того как семейство протоколов TCP/IP было реализовано и внедрено, Международная организация по стандартизации (International Organization for Standardization — ISO) предложила собственную семиуровневую сетевую модель, названую OSI (Open System Interconnection — модель взаимодействия открытых систем). Она так никогда и не приобрела широкой популярности из-за своей сложности и неэффективности. Некоторые аналитики даже считают, что к семи уровням OSI нужно добавить еще два: финансовый и политический[3].

На рис. 12.1 изображена многоуровневая схема взаимодействия различных компонентов и клиентов TCP/IP.

[2] Мы немного грешим против истины, утверждая, что протокол ARP входит в семейство TCP/IP. На самом деле он вполне может использоваться вместе с другими семействами протоколов. Просто этот протокол является неотъемлемой частью стека TCP/IP в большинстве локальных сетей.

[3] Со схемой этой расширенной девятиуровневой модели можно ознакомиться на Web-сайте компании Internet Systems Consortium `www.isc.org`.

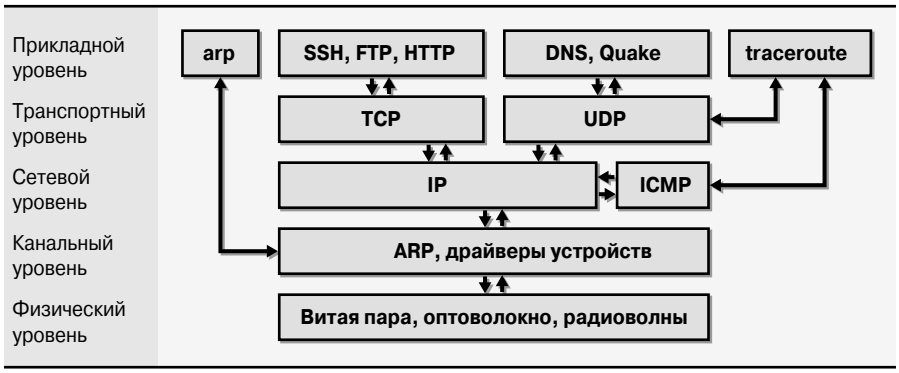


Рис. 12.1. Успешное семейство TCP/IP

12.3. Пакеты и их инкапсуляция

Linux располагает средствами поддержки целого ряда физических сетей, включая Ethernet (в том числе беспроводная связь), FDDI, Token Ring, ATM и системы с последовательными соединениями. Управление аппаратными устройствами осуществляется на канальном уровне стека TCP/IP, а протоколам более высоких уровней неизвестно, как именно используются аппаратные средства.

Данные передаются по сети в виде пакетов, которые имеют максимальный размер, определяемый ограничениями канального уровня. Каждый пакет состоит из заголовка и полезного содержимого. Заголовок содержит сведения о том, откуда прибыл пакет и куда он направляется. В него могут также включаться контрольные суммы, информация, характерная для конкретного протокола, и другие инструкции, касающиеся обработки пакета. Полезное содержимое — это данные, подлежащие пересылке.

Название базового блока передачи данных зависит от уровня протокола. На канальном уровне это *кадр*, или *фрейм*, в протоколе IP — *пакет*, а в протоколе TCP — *сегмент*. Мы будем придерживаться универсального термина "*пакет*".

Готовящийся к отправке пакет передается вниз по стеку протоколов, и каждый протокол добавляет в него собственный заголовок. Сформированный пакет одного протокола становится полезным содержимым пакета, генерируемого следующим протоколом. Эта операция называется *инкапсуляцией*. На принимающей стороне инкапсулированные пакеты восстанавливаются в обратном порядке при прохождении вверх по стеку.

Например, дейтаграмма (пакет UDP), передаваемая по сети Ethernet, упакована в трех различных "конвертах". В среде Ethernet она "вкладывается" в простой физический фрейм, заголовок которого содержит сведения об аппаратных адресах отправителя и ближайшего получателя, длине фрейма и его контрольной сумме (CRC). Полезным содержимым Ethernet-фрейма является IP-пакет. Полезное содержимое IP-пакета — это UDP-пакет, и, наконец, полезное содержимое UDP-пакета состоит из собственно данных, подлежащих передаче. Компоненты такого пакета изображены на рис. 12.2.

Под словом "байт" мы подразумеваем 8-битовую цепочку данных. Когда-то этот термин имел более широкое значение, поэтому иногда в документах RFC можно встретить термин "октет".

Рис. 12.2. Стандартный сетевой пакет

Канальный уровень

На канальном уровне обеспечивается связь между сетевым программным обеспечением и собственно сетевым оборудованием.

Стандарты формирования фреймов Ethernet

На канальном уровне к пакетам добавляются заголовки и вставляются разделители между ними. Заголовки содержат информацию об адресах канального уровня и контрольные суммы, а разделители позволяют принимающей стороне понять, где заканчивается один пакет и начинается другой. Процесс добавления вспомогательных битов называется формированием фреймов или кадровой разбивкой.

В настоящее время в десятимегабитных сетях Ethernet используется единственный стандарт формирования фреймов: DIX Ethernet II[4]. Исторически сложилось так, что сетях Novell наибольшее распространение получил другой стандарт — IEEE 802.2 LLC/SNAP. В некоторых сетях все еще можно встретить следы применения этого стандарта формирования фреймов.

Выбор стандарта формирования фреймов диктуется имеющейся сетевой платой и ее драйвером. На персональных компьютерах, работающих под управлением Windows, выбор можно делать самостоятельно, а в Linux — как правило, нет. В Linux оба стандарта взаимозаменяемы. А вот в среде Windows компьютеры в одной сети, использующие различные стандарты, не могут общаться друг с другом. Системному администратору обычно не приходится заниматься вопросами формирования фреймов, если только не выполняется низкоуровневая отладка в смешанной сети.

Стандарты кабелей Ethernet

Выбор кабелей для различных вариантов Ethernet (10 Мбит/с, 100 Мбит/с, 1 Гбит/с и современный 10 Гбит/с) определяется соответствующими стандартами IEEE. Обычно при появлении новой технологии выпускают кабель с малым предельным расстоянием передачи. Позднее на рынке появляются более дешевые аналоги, рассчитанные на большие расстояния.

[4] Канальный уровень в действительности разбит на два подуровня: MAC (Media Access Control — управление доступом к среде) и LLC (Logical Link Control — управление логическим соединением). На подуровне MAC осуществляется перенос пакетов в кабельную систему. Подуровень LLC отвечает за формирование фреймов.

Дополнительные сведения о различных стандартах Ethernet приведены в главе 14. Другим полезным источником информации по этой теме является Web-сайт `www.ethermanage.com/ethernet/`, поддерживаемый Чарльзом Сперджоном (Charles Spurgeon).

Беспроводные сети

В спецификации IEEE 802.11 делается попытка описать стандарты формирования фреймов и передачи сигналов для беспроводных сетей. К сожалению, спецификация весьма расплывчата и включает ряд параметров, которые не полностью определены. При взаимодействии разнородных сетей приходится учитывать такие аспекты, как трансляция и инкапсуляция.

В случае трансляции пакет преобразуется из одной формы в другую, а при инкапсуляции он упаковывается в структуру нужного формата. По умолчанию в среде Windows обычно применяется инкапсуляция, а в Linux — трансляция, поэтому базовые беспроводные станции должны конфигурироваться явным образом. Когда развертывается беспроводная сеть, нужно убедиться в том, что базовая станция и связанные с ней рабочие станции функционируют в одинаковом режиме.

Максимальная единица передачи

Размер сетевых пакетов ограничивается как характеристиками аппаратных средств, так и требованиями протоколов. Например, объем полезного содержимого стандартного Ethernet-фрейма не может превышать 1500 байт. Предельный размер пакета устанавливается на канальном уровне и называется максимальной единицей передачи (Maximum Transfer Unit, MTU). Стандартные значения параметра MTU для разных видов сетей приведены в табл. 12.2.

Таблица 12.2. Максимальные размеры передаваемых блоков в сетях различных типов

Тип сетевого соединения	Максимальная единица передачи
Ethernet	1500 байт (1492 в спецификации 802.2)
FDDI	4470 байт (4352 для IP/FDDI)
Token Ring	Конфигурируется [a]
Модемное PPP-соединение	Конфигурируется, обычно 512 или 576 байт
Стеки протоколов персональных компьютеров	Конфигурируется, по умолчанию обычно 512 байт
Двухточечные каналы ГВС (T1, T3)	Конфигурируется, обычно 1500 или 4500 байт

[a] Распространенные значения таковы: 552, 1064, 2088, 4508 и 8232. Иногда выбирается значение 1500 для совместимости с Ethernet.

В семействе TCP/IP протокол IP отвечает за разбивку пакета на такие фрагменты, чтобы их размер соответствовал требованиям конкретного сетевого соединения. Если пакет проходит через несколько сетей, в одной из них параметр MTU может оказаться меньшим, чем в исходной сети. В этом случае маршрутизатор, пересылающий пакет в сеть с меньшим значением MTU, подвергнет пакет дальнейшей фрагментации. Подобный процесс нежелателен в условиях сильной загруженности маршрутизаторов.

Протокол TCP способен определить наименьшее значение MTU вдоль всего пути следования пакета и с самого начала разбить пакет в соответствии с этим значением. Протокол UDP не столь “интеллектуален” и перекладывает выполнение этой задачи на

протокол IP. В стандарте IPv6 промежуточные маршрутизаторы больше не могут выполнять фрагментацию пакетов, а режим определения максимального размера блока является обязательным.

Иногда проблема фрагментации оказывается достаточно коварной. Хотя в процессе определения MTU для выбранного маршрута все MTU-конфликты должны автоматически разрешаться, иногда требуется вмешательство администратора. Например, в виртуальной частной сети с туннельной структурой необходимо проверять размер пакетов, проходящих через туннель. Обычно их начальный размер — 1500 байт, но когда к ним добавляется туннельный заголовок, размер пакетов становится равным примерно 1540 байт, и уже требуется фрагментация. Уменьшение максимального размера передаваемого блока позволяет избежать фрагментации и повысить производительность туннельной сети. Просмотрите `man`-страницу команды `ifconfig`, чтобы узнать, как настроить параметр MTU сетевой платы.

Адресация пакетов

Подобно письмам и сообщениям электронной почты, сетевые пакеты могут достичь пункта назначения только при наличии правильного адреса. В TCP/IP используется сочетание нескольких схем адресации:

- MAC-адресов сетевого оборудования;
- IP-адресов программного обеспечения;
- текстовых имен компьютеров.

В соответствии с этим сетевая плата имеет MAC-адрес канального уровня, который отличает ее от других сетевых плат физической сети, IP-адрес, определяющий ее положение в сети Internet, и текстовое имя, понятное пользователям.

Самый нижний уровень адресации задается сетевыми аппаратными средствами. Например, Ethernet-устройствам в процессе изготовления назначаются уникальные шестибайтные аппаратные адреса[5]. Сетевые платы Token Ring также имеют шестибайтные адреса. В некоторых сетях с двухточечным соединением (например, в PPP-сетях, которые описываются в разделе 13.13) аппаратные адреса вообще не нужны: адрес пункта назначения указывается непосредственно при установке соединения.

Шестибайтный Ethernet-адрес разбивается на две части: первые три байта определяют изготовителя устройства, а последние три — выступают в качестве уникального серийного номера, назначаемого изготовителем. Системные администраторы могут выяснить марку устройства, вызывающего проблемы в сети, поискав трехбайтовый идентификатор соответствующих пакетов в таблице идентификаторов изготовителей. Текущая таблица доступна по адресу `www.iana.org/assignments/ethernet-numbers`.

В свое время эта информация регулярно публиковалась в виде документов RFC, но затем такая практика прекратилась. Последним в серии *Assigned Numbers* (назначенные номера) был документ RFC1700 (1994 г.). Официальным хранилищем всех специальных имен и номеров, принятых в сети Internet, является Web-страничка `www.iana.org/numbers.htm`.

[5] Уникальность эта теоретическая. Однажды компания 3Com продублировала Ethernet-адреса в платах, предназначенных для разных типов сетей. Разработчики “разумно” предположили, что заказчики будут покупать платы только одного типа. Следствием подобного рационализаторства стал хаос в некоторых организациях, где использовались сети смешанного типа, и даже во внутренней сети компании 3Com. Конфликты MAC-адресов являются фатальными в пределах одной сети, но некритичными в сетях, разделенных маршрутизатором.

Аппаратные адреса Ethernet должны назначаться на постоянной основе и оставаться неизменными. К сожалению, некоторые сетевые платы допускают программное задание аппаратных адресов. Особенно сложно в этом отношении с платами беспроводной связи. Избегайте назначения адресов из диапазона группового вещания (вторая цифра — нечетная) и других специальных адресов.

В Linux можно менять аппаратный адрес платы Ethernet, но лучше этого не делать, так как не все брандмауэры и DHCP-серверы допускают такую замену. Впрочем, подобный прием оказывается весьма кстати, если необходимо заменить вышедший из строя компьютер или сетевую плату, но по какой-то причине MAC-адрес должен остаться прежним (например, его фильтруют все коммутаторы или DHCP-сервер выделяет IP-адреса на основании MAC-адресов).

На следующем, более высоком уровне используется Internet-адресация (чаще называемая IP-адресацией). Каждому сетевому интерфейсу здесь присваивается один[6] четырехбайтовый IP-адрес. Эти адреса глобально уникальны[7] и аппаратно независимы. Мы уделим им достаточно много внимания в разделе 12.4.

📖 Подробнее о протоколе ARP рассказывается в разделе 12.6.

Соответствие между IP-адресами и аппаратными адресами устанавливается на канальном уровне модели TCP/IP. В сетях, поддерживающих широковещательный режим (т.е. в сетях, позволяющих адресовать пакеты “всем компьютерам данного физического сегмента”), протокол ARP обеспечивает автоматическую привязку адресов без вмешательства системного администратора.

Поскольку IP-адреса представляют собой длинные, на первый взгляд случайные числа, запоминать их трудно. Linux-системы позволяют закреплять за IP-адресом одно или несколько текстовых имен, чтобы вместо 216.115.108.245 пользователь мог ввести “www.yahoo.com”.Существует несколько способов подобной привязки: с помощью статического файла (`/etc/hosts`), базы данных NIS и, наконец, DNS — глобальной системы доменных имен. Помните о том, что имя компьютера — это лишь сокращенный способ записи IP-адреса и не более того.

Порты

IP-адреса идентифицируют компьютеры, или, точнее говоря, сетевые интерфейсы компьютеров. Они недостаточно подробны, чтобы можно было адресовать конкретные процессы или службы. Протоколы TCP и UDP расширяют концепцию IP-адресов, вводя понятие порта. Порт представляет собой 16-разрядное число, добавляемое к IP-адресу и указывающее конкретный канал взаимодействия. Всем стандартным службам, в частности электронной почте, FTP и WWW, назначаются порты, которые определены в файле `/etc/services`. Для того чтобы предотвратить попытки посторонних процессов замаскироваться под стандартные службы, Linux-системы предоставляют доступ к портам с номерами до 1024 только процессам пользователя `root` (впрочем, это ограничение не распространяется на доступ других пользователей к самим привилегированным процессам).

---

[6] Вообще-то, с сетевым интерфейсом может быть связано несколько IP-адресов, но это особая конфигурация, применяемая лишь в специальных случаях. Подробнее об этом рассказывается в разделе 21.3.

[7] Это не всегда верно. В разделе 12.4, когда речь пойдет о системе NAT, будет рассказано, в каких случаях адреса теряют свою уникальность.

Типы адресов

В протоколе IP и на канальном уровне поддерживается несколько типов адресов:

- направленные (unicast) — адреса, которые обозначают отдельный компьютер (в действительности сетевой интерфейс);
- групповые (multicast) — адреса, идентифицирующие группу хостов;
- широковещательные (broadcast) — адреса, обозначающие все хосты локальной сети.

Режим группового вещания используется, к примеру, в видеоконференциях, где одна и та же последовательность пакетов посылается всем участникам конференции. Протокол IGMP (Internet Group Management Protocol — протокол управления группами хостов Internet) отвечает за управление совокупностями хостов, идентифицируемыми как один обобщенный адресат. В протоколе IP групповые адреса начинаются с байта, значение которого лежит в диапазоне 224–239.

В широковещательных адресах последняя часть адреса в двоичном виде состоит из единиц.

В настоящее время в Internet групповые адреса практически не используются, но их применению посвящено несколько ведущихся исследовательских проектов. Подробнее о них можно узнать на сайте `Internet2.org`.

12.4. IP-адреса

IP-адрес состоит из четырех байтов и разбит на две части: сетевую и машинную. Первая часть обозначает логическую сеть, к которой относится адрес, а вторая идентифицирует конкретный компьютер этой сети.

По соглашению IP-адреса записываются в виде группы десятичных чисел (по одному на каждый байт), разделенных точками. Например, IP-адрес нашего компьютера `boulder` записывается как 128.138.240.1. Самый левый байт — старший; он всегда относится к сетевой части адреса.

Если первым байтом адреса является число 127, то оно обозначает *интерфейс обратной связи* — фиктивную сеть, не имеющую реального аппаратного интерфейса и состоящую из одного компьютера. Адрес 127.0.0.1 всегда ссылается на текущий компьютер. Ему соответствует символическое имя *`localhost`*.

IP-адрес и другие параметры сетевого интерфейса задаются командой `ifconfig`. Ее описание приведено в разделе 12.7.

Классы IP-адресов

Исторически IP-адреса группировались в классы, определявшиеся на основании первых битов самого левого байта. Классы отличались распределением байтов адреса между сетевой и машинной частями. Современные маршрутизаторы используют явные маски для задания сетевой части адреса, причем компоненты адреса могут разделяться не обязательно по границе байтов. Тем не менее, традиционные классы все еще используются по умолчанию, если не указана явная маска.

Классы A, B и C обозначают обычные IP-адреса, а классы D и E применяются при групповой адресации и в исследовательских целях. В табл. 12.3 представлены характеристики каждого класса адресов. Сетевая часть адреса помечена буквой C, а машинная — буквой M.

Таблица 12.3. Классы IP-адресов

Класс	Первый байт[a]	Формат	Комментарии
A	1-126	C.M.M.M	Самые первые сети или адреса, зарезервированные для Министерства обороны США
B	128-191	C.C.M.M	Крупные организации, обычно с подсетями; адреса данного класса почти полностью заняты
C	192-223	C.C.C.M	Небольшие организации; адреса данного класса получить легко, они выделяются целыми блоками
D	224-239	–	Групповые адреса, не назначаются на постоянной основе
E	240-255	–	Экспериментальные адреса

[a] Значение 0 не используется в качестве первого байта обычных IP-адресов. Значение 127 зарезервировано для адресов обратной связи.

Подсети и маски

В редких случаях в состав локальной сети входит более ста компьютеров. По этой причине полезность адресов класса A или B (которые допускают наличие в одной сети, соответственно, 16 777 214 и 65 534 хостов) весьма сомнительна. К примеру, 126 адресов класса A занимают половину доступного адресного пространства!

В большинстве организаций, имеющих эти адреса, пользуются улучшенным вариантом схемы адресации, в которой применяются подсети. В данном случае некоторая доля машинной части адреса "заимствуется" для расширения сетевой части. Например, четыре байта адреса класса B обычно интерпретируются как C.C.M.M. При наличии подсетей третий байт также отводится под номер сети, а не номер хоста, поэтому формат адреса становится таким: C.C.C.M. Это превращает одну сеть класса B в 256 подсетей, подобных сетям класса C, т.е. в каждую из них может входить 254 компьютера.

🕮 Подробнее о команде `ifconfig` рассказывается в разделе 12.4.

Переназначение адресов осуществляется с помощью команды `ifconfig`, которая присваивает сетевому интерфейсу так называемую маску подсети. Каждый бит маски, соответствующий сетевой части IP-адреса, равен 1, а биты машинной части равны 0. Например, маска для конфигурации C.C.C.M будет иметь вид 255.255.255.0 в десятичной системе и 0xFFFFFF00 — в шестнадцатеричной. Команда `ifconfig`, как правило, использует класс адреса для того, чтобы выяснить, какие биты относятся к сетевой части. Если задается явная маска, то эта функция просто отменяется.

Граница между сетевой и машинной частями адреса не обязательно приходится на границу байта. Биты сетевой части должны быть смежными и располагаться начиная со старшего байта[8].

Сетевые маски, не оканчивающиеся на границе байта, труднее декодировать. Они обычно записываются в виде суффикса */XX*, где *XX* — число битов в сетевой части адреса (длина маски). Такую запись иногда называют нотацией CIDR (Classless Inter-Domain Routing — протокол бесклассовой междоменной маршрутизации). Например, адрес 128.138.243.0/26 обозначает первую из четырех сетей с общим компонентом адреса 128.138.243. В трех других сетях последний байт адреса равен 64, 128 и 192. Сетевая мас-

[8] Конфигурация вида C.C.M.C раньше допускалась, хотя и не была распространена. Сегодня подобный формат адреса недопустим.

ка, связанная с этими сетями, имеет вид 255.255.255.192, или 0xFFFFFFC0. В двоичном представлении это 26 единиц с последующими шестью нулями (рис. 12.3).

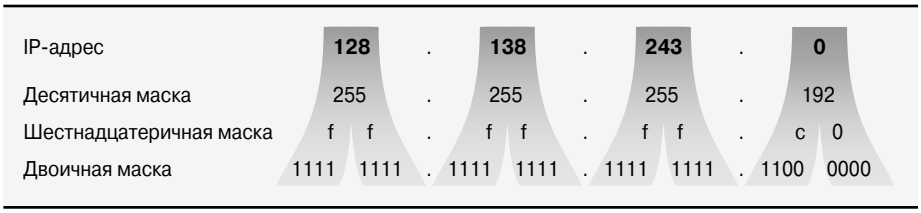


Рис. 12.3. Структура маски подсети в разных системах счисления

В сети с маской /26 для нумерации хостов отводится шесть битов (32 – 26 = 6). Таким образом, появляется возможность задать 64 адреса ($2^6 = 64$). В действительности допускается использовать лишь 62 адреса, поскольку адреса, полностью состоящие из нулей или единиц, зарезервированы (соответственно, для самой сети и широковещательного режима).

Манипулировать всеми этими битами в уме трудно, но есть ряд приемов, позволяющих упростить вычисления. Число хостов в сети и значение последнего байта сетевой маски в сумме всегда дают 256: последний байт сетевой маски = 256 — размер сети.

Так, в рассмотренном выше случае формула даст результат 256 – 64 = 192, где 192 — последний байт маски. Другое правило гласит о том, что значение последнего байта фактического адреса сети (не сетевой маски) должно нацело делиться на число хостов сети. В рассматриваемом примере последние байты равны 0, 64, 128 и 192 — каждое из этих чисел делится нацело на 64.

В нашем примере старшие два бита последнего байта адреса могут принимать значения 00, 01, 10 и 11. Таким образом, сеть 128.138.243.0/24 может быть разделена на четыре сети /26:

- 128.138.243.0/26 (0 — **00**000000);
- 128.138.243.64/26 (64 — **01**000000);
- 128.138.243.128/26 (128 — **10**000000);
- 128.138.243.192/26 (192 — **11**000000);

Выделенные полужирным шрифтом биты последнего байта каждого адреса относятся к сетевой части.

Имея только IP-адрес (допустим, 128.138.243.100), невозможно сказать, каким будет адрес сети и широковещательный адрес. В табл. 12.4 представлены возможные варианты для масок /16 (выбирается по умолчанию для адресов класса B), /24 и /26 (наиболее приемлемая длина, если имеющееся адресное пространство невелико).

Таблица 12.4. Пример расшифровки IP-адреса

IP-адрес	Сетевая маска	Адрес сети	Широковещательный адрес
128.138.243.100/16	255.255.0.0	128.138.0.0	128.138.255.255
128.138.243.100/24	255.255.255.0	128.138.243.0	128.138.243.255
128.138.243.100/26	255.255.255.192	128.138.243.64	128.138.243.127

Удобный сетевой сценарий IP Calculator, созданный Крисчен Джодиес (Krischan Jodies) (`www.jodies.de/ipcalc`) помогает выполнять адресную арифметику. Сценарий отображает всю информацию, касающуюся заданного сетевого адреса и его маски. Кроме сетевой версии существует также утилита `tarball`, а дистрибутивы Debian и Ubuntu содержат версию команду `ipcalc`, запускаемую из командной строки.

Ниже приведен образец работы сценария (формат вывода для наглядности немного изменен):

```
Address:   24.8.175.69          00011000.00001000.10101111.01000101
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111.00000000
Wildcard:  0.0.0.255            00000000.00000000.00000000.11111111
=>
Network:   24.8.175.0/24        00011000.00001000.10101111.00000000(Class A)
Broadcast: 24.8.175.255         00011000.00001000.10101111.11111111
HostMin:   24.8.175.1           00011000.00001000.10101111.00000001
HostMax:   24.8.175.254         00011000.00001000.10101111.11111110
```

Вывод содержит как общепринятые десятичные, так двоичные, пригодные для непосредственной вставки, версии адресов. Это очень удобно.

В Red Hat имеется утилита, тоже называющаяся `ipcalc`, но она довольно неуклюжа и в большинстве случаев предполагает, что IP-адреса относятся к классу A, B или C.

Исходный RFC-документ, посвященный IP-подсетям (RFC950), не разрешал использовать первый и последний номера подсети (их компоненты, относящиеся к сетевой части адреса, состоят из всех нулей и всех единиц, соответственно). В нашем случае это привело бы к исключению половины подсетей: с номерами 0 и 192. В действительности все игнорировали это правило, кроме компаний Novell и Cisco. (В первых версиях операционной системы IOS компании Cisco приходилось явно разрешать подсеть 0 с помощью команды `ip subnet zero`. В версиях 12.0 и выше подсеть 0 доступна по умолчанию.)

Авторы документа ошиблись, хотя их намерения были благородны. Запрет подсети 0 был вызван опасением, что возникнет путаница из-за совпадения адреса подсети и адреса самой сети без маски. Этот страх оказался беспочвенным, поэтому адреса подсетей, сетевые компоненты которых состоят из всех нулей или всех единиц, широко используются сегодня. На самом деле подобный запрет должен относиться к машинной части адреса.

Сетевой (все нули в машинной части) и широковещательный (все единицы в машинной части) адреса сокращают число хостов в каждой локальной сети на 2, поэтому в самой маленькой сети формально должно быть 4 хоста: два реальных, соединенных напрямую, и два фиктивных (сетевой и широковещательный). Для того чтобы создать такую сеть, необходимо отвести два бита под машинную часть адреса, т.е. суффикс адреса будет /30, а сетевая маска — 255.255.255.252, или 0xFFFFFFFC. Но есть еще сеть с маской /31, которая трактуется как особый случай (RFC3021): у нее нет сетевого и широковещательного адресов, а оба оставшихся адреса используются для идентификации хостов. Маска такой сети равна 255.255.255.254.

Сетевые маски понятны компьютерам локальной сети, но внешний мир о них не знает и интерпретирует адреса в соответствии с заложенными в них классами[9]. В нашем случае (адрес 128.138.243.100) не нужно сообщать миру о каждой подсети: достаточно

[9] Это довольно упрощенное описание, так как нужно еще учитывать влияние протокола CIDR.

сообщить лишь об одной сети класса B. Когда пакет прибывает в пункт назначения, после которого происходит разделение на подсети, к адресу применяется локальная сетевая маска, выявляется реальный получатель, и пакет направляется ему.

Кризис IP-адресации

Примерно в 1992 г. сообщество Internet наконец-то осознало три фундаментальные проблемы, возникающие вследствие применения исходной схемы выделения адресов.

- Во-первых, при существующих темпах развития глобальной сети адресное пространство класса B — наиболее желанное для относительно крупных организаций — должно было исчерпаться к середине 1995 г.
- Во-вторых, таблицы маршрутизации серверов, управляющих Internet-магистралями, стали настолько большими, что возникли опасения в нехватке памяти на маршрутизаторах.
- Наконец, в-третьих, IP-адреса выделялись по принципу "первым попросил — первым получил", без учета информации о местоположении адресата. Таким образом, "соседние" адресаты могли находиться как в одной организации, так и на разных континентах.

Представьте, какая путаница могла возникнуть, если бы так же бессистемно назначались почтовые индексы!

Для решения проблемы были одновременно предложены два подхода: один на ближайшее будущее, а другой — более долгосрочный. Первое из решений — упоминавшийся выше протокол CIDR — предусматривало изменение принципов управления существующими четырехбайтовыми адресами и упрощение таблиц маршрутизации за счет введения понятия смежности.

Долговременным решением является стандарт IPv6. Он представляет собой следующее поколение протокола IP, в котором длина адреса увеличена до 16 байт и учтен ряд других ошибок, выявленных на протяжении 25 лет эксплуатации протокола IP. Некоторые старые элементы вообще исключены из нового стандарта, так как опыт показал, что они не имеют практической ценности. В результате полученный протокол стал потенциально быстрее и проще в реализации. В нем интегрированы средства аутентификации и обеспечения безопасности, а также устранена процедура фрагментации пакетов на промежуточных маршрутизаторах.

Протокол IPv6 все еще проходит начальные этапы внедрения. А вот протокол CIDR эксплуатируется уже не один год; он поддерживается серверами Internet-магистралей и ведущими производителями маршрутизаторов. Существенную роль в уменьшении требуемого числа IP-адресов сыграла также система NAT (Network Address Translation — трансляция сетевых адресов), обеспечивающая повторное использование существующих адресов (описывается в разделе 12.4).

Сложность стандарта IPv6, эффективность протокола CIDR и системы NAT, а также инертность существующей сети Internet заставляют предполагать, что потребуется некоторое время, прежде чем произойдет переход на IPv6. Хотя многие приложения и операционные системы имеют встроенную поддержку IPv6, затраты, связанные с внедрением этого стандарта все еще удерживают многие организации США от его активного использования. Тем не менее, значительные усилия, прикладываемые в этом направлении частью компаний-поставщиков программного обеспечения и независимыми разработчиками, не оставляют сомнения в том, что в нужный момент система Linux будет вполне готова к переходу на IPv6.

На международной арене проблема адресации IPv4 ощущается более остро, поэтому здесь стандарт IPv6 был встречен с большим энтузиазмом. В США переход к IPv6 может быть ускорен каким-нибудь ориентированным на него суперпопулярным приложением, например новым поколением мобильных телефонов, которые ставят адрес IPv6 в соответствие номеру телефона. Системы передачи голосовых данных по IP-сетям тоже выиграют от приведения в соответствие телефонных номеров и адресов IPv6.

Дополнительные детали адресации в стандарте IPv6 приведены в подразделе "Адресация IPv6".

CIDR: протокол бесклассовой междоменной маршрутизации

Протокол CIDR, описанный в документе RFC1519 (сентябрь 1993 г.), устраняет потребность в системе классов, на основании которой раньше определялась сетевая часть IP-адреса. Он является прямым расширением идеи подсетей, поскольку тоже позволяет задавать сетевую маску, определяющую границу между сетевой и машинной частями адреса. Но в целях маршрутизации допускается, чтобы сетевая часть была *меньше*, чем того требует класс адреса. Благодаря укороченной маске возникает эффект объединения нескольких сетей. По этой причине протокол CIDR иногда называют протоколом формирования суперсетей.

При использовании протокола CIDR можно одной организации выделить несколько сетей класса C, причем не нужно для каждой из них создавать отдельную запись в таблице маршрутизации. Организации можно также выделить подпространство адресов класса A или B. Предположим, организации предоставлен блок из восьми адресов класса C, пронумерованных последовательно от 192.144.0.0 до 192.144.7.0 (в нотации CIDR —192.144.0.0/21). Внутри самой организации адреса могут распределяться так:

- 1 сеть с длиной маски 21 — 2046 хостов[10], сетевая маска 255.255.248.0;
- 8 сетей с длиной маски 24 — 254 хоста в каждой, сетевая маска 255.255.255.0;
- 16 сетей с длиной маски 24 — 126 хостов в каждой, сетевая маска 255.255.255.128;
- 32 сети с длиной маски 26 — 62 хоста в каждой, сетевая маска 255.255.255.192;

и т.д. Допускается смешение подсетей с разной длиной маски, если только они не перекрывают друг друга. Это называется формированием подсетей переменного размера. Например, провайдер Internet, которому выделена адресная область 193.143.0.0/21, может создать группу сетей /30 для коммутируемых PPP-клиентов, несколько сетей /24 для крупных клиентов и ряд сетей /27 для более мелких компаний.

Все хосты конкретной сети должны конфигурироваться с помощью одной сетевой маски. Нельзя для одного хоста задать длину маски 24, а для другого — 25.

Ценность протокола CIDR заключается в том, что для перечисленных адресов не обязательно иметь 256, 128 или даже 32 записи в таблице маршрутизации. Ведь все они ссылаются на хосты одной и той же организации, поэтому пакеты предварительно нужно доставлять в общий приемный пункт. А в таблицу маршрутизации достаточно внести запись 199.144.0.0/21. Протокол CIDR позволяет даже выделять фрагменты адресных пространств классов A и B, благодаря чему увеличивается число доступных адресов.

[10] Первоначальная спецификация Ethernet для коаксиального кабеля RG-RG-11 допускала наличие не более 1024 хостов в одной сети. Современные коммутаторы позволяют создавать сети гораздо большего размера, хотя это не слишком практично.

С появлением протокола CIDR системным администраторам пришлось поднатореть в двоичной и шестнадцатеричной арифметике. Правда, многие сумели открыть для себя Linux-утилиту `bc`, которая выполняет преобразования в любой системе счисления при помощи директив `ibase` и `obase`[11]. В табл. 12.5 представлена шпаргалка по нашему примеру.

Когда в 1993 г. появился протокол CIDR, таблицы маршрутизации магистральных серверов содержали примерно 20 000 записей. Медленный, но неуклонный рост сети Internet привел к тому, что к 2006 г. размер маршрутных таблиц увеличился до 250000 записей[12].

Таблица 12.5. Конфигурация сети в зависимости от длины сетевой маски

Длина маски[а]	Машинная часть	Хостов/сетей[б]	Сетевая маска (десятичная)	Сетевая маска (шестнадцатеричная)
/20	12	4094	255.255.240.0	0xFFFFF000
/21	11	2046	255.255.248.0	0xFFFFF800
/22	10	1022	255.255.252.0	0xFFFFFC00
/23	9	510	255.255.254.0	0xFFFFFE00
/24	8	254	255.255.255.0	0xFFFFFF00
/25	7	126	255.255.255.128	0xFFFFFF80
/26	6	62	255.255.255.192	0xFFFFFFC0
/27	5	30	255.255.255.224	0xFFFFFFE0
/28	4	14	255.255.255.240	0xFFFFFFF0
/29	3	6	255.255.255.248	0xFFFFFFF8
/30	2	2	255.255.255.252	0xFFFFFFFC

[а] Размеры сетевой и машинной частей адреса в сумме всегда дают 32. Такова разрядность IP-адресов.

[б] Число хостов в сети определяется по формуле 2(длина машинной части) — 2. Адреса хостов, машинная часть которых состоит из всех нулей или всех единиц, являются зарезервированными.

Существует одна неагрегированная область адресного пространства, называемая "болотом 192" (есть также ряд более мелких "болот" в адресных диапазонах 199 и 205). Она состоит из первых адресов класса C, владельцы которых не могут выполнить их агрегацию и не хотят поменять их на другие адреса. Особенно остро эта проблема стоит в США. Европа и Азия, начавшие подключаться к Internet позже, учли ошибки США и более разумно подошли к вопросу выделения адресов. Организации, имеющие неагрегированные адреса в диапазоне 192, должны отменить их регистрацию в Американском реестре номеров Internet (American Registry for Internet Numbers, ARIN) и получить от своих провайдеров новые блоки адресов. К сожалению, стоимость перерегистрации (по

[11] Но при ее использовании будьте внимательны. Мы предлагаем читателям самостоятельно разобраться в ее применении.

[12] Последнюю информацию о таблице BGP-маршрутизации можно найти на сайте `bgp.potaroo.net`.

крайней мере, в адресном пространстве IPv4) отпугивает большинство организаций от подобного шага.

Хотя протокол CIDR предусматривался как промежуточное решение, оказалось, что он отлично справляется с проблемой роста сети Internet в обозримом будущем. В сочетании с NAT он зарекомендовал себя настолько хорошо, что теперь неясно, нужен ли вообще новый протокол IP. Конечно, в разработку и коммерческую реализацию спецификации IPv6 был вложен огромный инженерный труд, и будет стыдно, если он окажется напрасным, но широкомасштабное внедрение стандарта IPv6, очевидно, напрямую зависит от появления какого-нибудь суперпопулярного приложения, ориентированного исключительно на IPv6, или от решения компании Microsoft отправить протокол IPv4 "на пенсию".

Выделение адресов

На заре эпохи Internet для получения адресов приходилось обращаться в Информационный центр сети Internet (Internet Network Information Center, InterNIC). В настоящее время в Америке ему эту функцию выполняют ARIN и LACNIC. Только провайдеры, выделяющие достаточно большое число адресов в год, имеют право направлять запросы в ARIN напрямую. Все остальные организации должны обращаться к своим региональным провайдерам.

Формально назначаются лишь адреса сетей. Организации должны самостоятельно определять номера компьютеров и закреплять за ними IP-адреса. Деление на подсети также осуществляется произвольно.

Организация ICANN в административном порядке делегировала полномочия по распределению адресов пяти региональным организациям, которые предоставляют блоки адресов провайдерам Internet в рамках своих регионов (табл. 12.6). Провайдеры, в свою очередь, выделяют адреса отдельным клиентам. Только крупные провайдеры могут напрямую посылать запросы в свои региональные реестры.

Таблица 12.6. Региональные реестры Internet-адресов

Организация	Web-адрес	Охватываемый регион
ARIN	`www.arin.net`	Северная Америка, часть Карибского бассейна
APNIC	`www.apnic.net`	Азия и Океания
AfriNIC	`www.afrinic.net`	Африка
LACNIC	`www.lacnic.net`	Центральная и Южная Америка, часть Карибского бассейна
RIPE	`www.ripe.net`	Европа и прилегающие регионы

Делегирование полномочий от ICANN к региональным реестрам, а затем к национальным или региональным провайдерам Internet обеспечивает должную агрегацию адресов в таблицах маршрутизации магистральных сетей. Клиенты, получившие адреса от своего провайдера, не имеют маршрутных записей в магистральных серверах. Достаточно наличия одной записи, ссылающейся на провайдера.

С самого начала адресное пространство было распределено не самым честным образом. Правительство США зарезервировало примерно половину адресов для себя, предоставив Европе и Азии сравнительно небольшие блоки. Однако в Европе и Азии гораз-

до рациональнее управляли своим адресным пространством, чем в Америке. Это очень хорошо видно на текущей карте распределения адресов. Она доступна по адресу

```
www.caida.org/analysis/learn/ipv4space
```

На этой карте демонстрируется все адресное пространство в целом, выделенные области, маршрутизируемые (т.е. доступные) области, а также адреса, трафик для которых постоянно замеряется в нескольких основных точках обмена трафиком в США.

Частные адреса и система NAT

Другое временное решение проблемы сокращающегося адресного пространства заключается в использовании частных областей IP-адресов, описанных в документе RFC1918 (февраль 1996 г.). Сегодня, в эпоху протокола CIDR, организации получают IP-адреса у провайдеров Internet. Если организация хочет сменить провайдера, она должна оплатить услуги по переадресации своих сетей. Это объясняется тем, что провайдеры предоставляют адресное пространство только своим клиентам. При смене провайдера можно попытаться убедить старого провайдера оставить зарезервированные за организацией адреса и попросить нового провайдера настроить на них систему маршрутизации. Но обычно провайдеры не хотят заниматься подобными вещами и требуют от клиентов перерегистрации.

Альтернативой использованию выделенных провайдером адресов являются частные адреса, которые неизвестны провайдеру. Документ RFC1918 определяет, что одна сеть класса A, 16 сетей класса B и 256 сетей класса C резервируются для частного использования и никогда не выделяются глобально. Суть в том, чтобы пакеты, несущие в себе частные адреса, никогда не выходили в глобальную сеть. Их должен фильтровать пограничный маршрутизатор. Если обнаруживается “утечка” пакетов, нужно выяснить, где была допущена ошибка конфигурации.

В табл. 12.7 показаны диапазоны частных адресов (в последней колонке каждый диапазон представлен в более короткой нотации протокола CIDR).

Таблица 12.7. IP-адреса, зарезервированные для частного использования

Класс	Начало	Конец	Диапазон CIDR
Class A	10.0.0.0	10.255.255.255	10.0.0.0/8
Class B	172.16.0.0	172.31.255.255	172.16.0.0/12
Class C	192.168.0.0	192.168.255.255	192.168.0.0/16

Из перечисленных диапазонов организации могут выбирать для себя сети нужного размера.

Чтобы хосты, использующие частные адреса, могли получать доступ в Internet, на пограничном маршрутизаторе организации должна выполняться система NAT (Network Address Translation — трансляция сетевых адресов). Эта система перехватывает пакеты и заменяет в них адрес отправителя реальным внешним IP-адресом. Может также происходить замена номера исходного порта. Система хранит таблицу преобразований между внутренними и внешними адресами/портами, чтобы ответные пакеты доставлялись нужному адресату.

Благодаря нумерации портов появляется возможность подключить несколько исходящих соединений к общему IP-адресу, чтобы внутренние хосты совместно использовали одинаковый внешний IP-адрес. Иногда в организации достаточно иметь один "настоящий" внешний адрес.

Организация, использующая систему NAT, запрашивает адреса у провайдера, но большинство адресов теперь используется для внутрисистемных привязок и не назначается отдельным компьютерам. Если организация захочет сменить провайдера, потребуется лишь изменить конфигурацию пограничного маршрутизатора и его системы NAT, но не самих компьютеров.

Система NAT широко используется для выделения адресов как отдельным клиентам, так и организациям. Можно также заставить Linux-систему выполнять функции NAT, хотя во многих организациях предпочитают делегировать эти обязанности маршрутизаторам или устройствам подключения к сети. По определенным причинам система NAT в Linux традиционно называлась "IP-маскированием", но с появлением ядра версии 2.4 разработчики перешли на более привычную терминологию.

Неправильная конфигурация системы NAT может привести к тому, что пакеты с частными адресами начнут проникать в Internet. Они достигнут хоста назначения, но ответные пакеты не смогут быть получены. Организация CAIDA[13], замеряющая трафик магистральных сетей, сообщает о том, что 0,1–0,2% пакетов, проходящих по магистрали, имеют либо частные адреса, либо неправильные контрольные суммы. На первый взгляд показатель кажется незначительным, но на самом деле это приводит к тому, что через хост MAE-West (одна из основных точек обмена трафиком) каждые 10 минут дополнительно проходит 20 000 пакетов. Информацию по статистике сети Internet и средствам измерения производительности глобальной сети можно получить на Web-сайте `www.caida.org`.

Одним из недостатков системы NAT (а возможно, и ее преимуществом) является то, что хост сети Internet не может напрямую подключиться к внутреннему компьютеру организации. В некоторых реализациях (например, в Linux и в брандмауэрах Cisco PIX) разрешается создавать туннели, которые поддерживают прямые соединения с выбранными хостами.

Другая проблема заключается в том, что некоторые приложения встраивают IP-адреса в информационную часть пакетов. Такие приложения не могут нормально работать совместно с NAT. К ним относятся определенные протоколы маршрутизации, программы потоковой доставки данных, такие как RealVideo и SHOUTcast, ряд FTP-команд, в частности PORT и PASV, система ICQ и многие игры. Это касается и виртуальных частных сетей.

В крупных корпорациях, где используются частные адреса и система NAT, требуется централизованная координация усилий, чтобы все компьютеры, независимо от офисной или административной принадлежности, имели уникальные IP-адреса. Проблемы возникают, когда одна компания приобретает другую компанию или сливается с ней. Если определенные сегменты их частных сетей перекрываются, приходится менять IP-адреса.

Система NAT скрывает внутреннюю структуру сети. Это может показаться достоинством с точки зрения безопасности, но специалисты считают, что на самом деле сис-

[13] CAIDA (Cooperative Association for Internet Data Analysis — совместная ассоциация по анализу данных в сети Internet) находится в Центре суперкомпьютеров, который расположен в здании Калифорнийского университета в Сан-Диего (`www.caida.org`).

тема NAT не обеспечивает должную безопасность и уж во всяком случае не устраняет потребность в брандмауэре. Кроме того, она препятствует попыткам оценить размеры и топологию сети Internet.

Была разработана, по меньшей мере, одна утилита, призванная идентифицировать хосты, которые используют систему NAT. Используемый в ней алгоритм сравнивает известные TTL-значения операционной системы с ожидаемыми значениями сети. Если они не совпадают, устройство с исходным IP-адресом считается NAT-устройством. Теоретически идея выглядит перспективной, но сопряжена с большими затратами и слишком громоздка для практической реализации. Подробные сведения об этом методе можно найти на сайте `www.sflow.org/detectNAT`.

Адресация в стандарте IPv6

В IPv6 адрес имеет длину 128 бит. Изначально столь длинные адреса вводились для того, чтобы решить проблему сокращающегося адресного пространства IPv4. Сегодня они служат также целям маршрутизации и локализации ссылок.

IP-адреса никогда не были географически сгруппированы подобно тому, как это имеет место в случае телефонных номеров или почтовых индексов. Теперь, с появлением в стандарте IPv6 понятия сегментации адресного пространства, IP-адреса стали группироваться в кластеры вокруг провайдеров Internet. Граница между сетевой и машинной частями адреса в IPv6 зафиксирована на отметке 64 бит. В сетевой части адреса граница между блоками общей и локальной топологии также зафиксирована и проходит на отметке 48 бит (табл. 12.8).

Таблица 12.8. Компоненты адреса IPv6

Полный адрес IPv6 (128 бит)

Префикс провайдера	Подсеть	Идентификатор хоста
45 бит	16 бит	64 бит

Тип адреса (3 бит)

Биты	Акроним	Интерпретация
1–3	FP	Format Prefix — префикс формата; определяет тип адреса, например направленный
4–16	TLA ID	Top-Level Aggregation ID — идентификатор агрегации верхнего уровня, например провайдера магистральной сети
17–24	RES	Reserved — зарезервировано на будущее
25–48	NLA ID	Next-Level Aggregation ID — идентификатор агрегации следующего уровня, например регионального провайдера Internet или конкретной организации
49–64	SLA ID	Site-Level Aggregation ID — идентификатор агрегации на уровне организации, например локальной подсети
65–128	INTERFACE ID	Идентификатор интерфейса (MAC-адрес плюс биты-заполнители)

Из перечисленных элементов только идентификаторы SLA и INTERFACE принадлежат хосту и организации, в которой он находится. Остальные идентификаторы предоставляются провайдером. Идентификатор SLA определяет локальную подсеть, а 64-разрядный идентификатор интерфейса — сетевую плату компьютера. В идентификаторе интерфейса, как правило, содержится 48-разрядный MAC-адрес, внутрь которого вставлены два байта-заполнителя (0xFFFE). Специальный бит MAC-адреса (седьмой слева в старшем байте), называемый битом "u", определяет, является адрес универсальным или локальным (RFC2373). Его наличие позволяет автоматически нумеровать хосты, что очень удобно для администраторов, которым остается управлять только подсетями.

В IPv6 MAC-адреса видны на уровне протокола IP, что имеет как положительные, так и отрицательные стороны. В частности, тип и модель сетевой платы кодируются в первой половине MAC-адреса, что облегчает задачу хакерам. Это вызвало тревогу у специалистов в области безопасности. Однако разработчики стандарта IPv6 указали на то, что использование MAC-адресов не является обязательным. Были предложены схемы включения случайного идентификатора в локальную часть адреса.

Но есть и положительный момент: назначать адреса IPv6 проще, чем адреса IPv4, так как нужно отслеживать только адрес подсети. Хосты могут самостоятельно конфигурировать себя — по крайней мере теоретически.

Префикс формата определяет тип адреса: направленный (unicast), групповой (multicast) или совмещенный (anycast). Для направленных адресов префикс равен 001 (в двоичной системе). Идентификаторы TLA и NLA определяют, соответственно, магистральный сервер верхнего уровня и локального провайдера Internet.

Большинство поставщиков оборудования готово к переходу на стандарт IPv6. Почти все дистрибутивы имеют встроенную поддержку IPv6, а сетевое оборудование поддерживает этот стандарт в течение уже почти десяти лет.

Как правило, ARIN выделяет адресное пространство IPv6 только для крупных провайдеров или для локальных Internet-реестров, которые планируют в ближайшем будущем использовать большие сегменты адресного пространства. Эти организации могут затем выделять подпространства своим клиентам. Минимальная арендная плата за использование такой адресной структуры составляет 1250 долларов в год, а максимальная — 36 000 долларов в год. ARIN практикует отказ от взимания платы с организаций с хорошей репутацией.

Вот несколько полезных источников информации, касающейся IPv6:

- `www.ipv6tf.net` — информационный портал IPv6;
- `www.ipv6.org` — списки часто задаваемых вопросов и различная техническая информация;
- `www.ipv6forum.com` — форум приверженцев IPv6.

Основное преимущество стандарта IPv6 заключается в том, что в нем есть средства, решающие проблему смены адресов. В пространстве IPv4 провайдеры выделяют адресные блоки своим клиентам, но эти адреса не являются переносимыми: когда клиент меняет провайдера, он должен вернуть старые адреса и запросить новые. В IPv6 новый провайдер просто предоставляет клиенту собственный адресный префикс, который добавляется к локальной части существующего адреса, возможно, даже на единственном компьютере: пограничном маршрутизаторе организации. Подобная схема напоминает систему NAT, но лишена ее недостатков.

Были предложены разные схемы перехода с IPv4 на IPv6, включая использование системы NAT для того, чтобы прятать адреса IPv6 при передаче пакетов по туннелю через существующие сети IPv4.

12.5. Маршрутизация

Маршрутизация — это процесс направления пакета по лабиринту сетей, находящихся между отправителем и получателем. В TCP/IP маршрутизация происходит примерно так, как путешественник, первый раз посетивший незнакомую страну, находит нужный ему дом, задавая вопросы местным жителям. Первый человек, с которым он заговорит, возможно, укажет ему нужный город. Войдя в город, путешественник спросит другого человека, и тот расскажет, как попасть на нужную улицу. В конце концов, наш путешественник подойдет достаточно близко к конечному пункту своих странствий, чтобы кто-нибудь указал ему дом, который он ищет.

Маршрутная информация в TCP/IP имеет форму правил (маршрутов), например: “Для того чтобы достичь сети A, посылайте пакеты через компьютер C”. Часто существует и стандартный маршрут. В нем объясняется, что нужно делать с пакетами, предназначенными для отправки в сеть, маршрут к которой не указан явным образом.

Данные маршрутизации хранятся в одной из таблиц ядра. Каждый элемент этой таблицы содержит несколько параметров, включая сетевую маску (раньше это поле было опциональным, но теперь оно обязательно, если стандартная сетевая маска неверна). Для направления пакета по заданному адресу ядро подбирает наиболее конкретный маршрут (т.е. тот, где самая длинная маска). Если ни один из маршрутов (в том числе стандартный) не подходит, то отправителю возвращается ICMP-сообщение вида “network unreachable” (сеть недоступна).

Слово “маршрутизация” широко употребляется в двух различных смыслах:

- оно означает процедуру поиска сетевого адреса в специальной таблице для передачи пакета в пункт его назначения;
- оно означает процесс построения самой такой таблицы.

Ниже мы рассмотрим, как осуществляется переадресация пакетов и как вручную добавить или удалить маршрут. Более сложные вопросы, связанные с работой протоколов маршрутизации, создающих и обслуживающих маршрутные таблицы, освещаются в главе 13.

Таблицы маршрутизации

Таблицу маршрутизации можно просмотреть с помощью команды `netstat -r`. Команда `netstat -rn` запрещает поиск доменных имен в DNS, вследствие чего все адреса будут представлены в числовом виде. Команда `netstat` подробно описывается в разделе 19.4, здесь же мы дадим небольшой пример, чтобы у читателей сложилось представление о том, что такое маршруты. В рассматриваемой системе есть два сетевых интерфейса: 132.236.227.93 (`eth0`) в сети 132.236.227.0/24 и 132.236.212.1 (`eth1`) в сети 132.236.212.0/26.

```
$ netstat -rn
Kernel IP routing table
Destination     Genmask          Gateway          Fl   MSS   Iface
132.236.227.0   255.255.255.0    132.236.227.93   U    1500  eth0
```

```
default         0.0.0.0          132.236.227.1     UG   1500  eth0
132.236.212.0   255.255.255.192  132.236.212.1     U    1500  eth1
132.236.220.64  255.255.255.192  132.236.212.6     UG   1500  eth1
127.0.0.1       255.255.255.255  127.0.0.1         U    3584  lo
```

Поле `destination` обычно содержит адрес сети. В поле `gateway` отображается адрес хоста. Например, четвертый маршрут указывает, что для достижения сети 132.236.220.64/26 пакеты следует посылать в шлюз 132.236.212.6 через интерфейс `eth1`. Вторая запись содержит описание стандартного маршрута. Пакеты, не адресованные явно ни в одну из трех указанных сетей (или самому компьютеру), будут направлены в стандартный шлюз 132.236.227.1. Хосты могут отправлять пакеты только тем шлюзовым компьютерам, которые непосредственно подключены к их сети.

📖 Полное описание команды `route` приведено в разделе 12.5.

Вести таблицы маршрутизации можно статически, динамически или комбинированным способом. Статический маршрут задается в явном виде с помощью команды `route`. Он должен оставаться в таблице маршрутизации в течение всего времени работы системы. Во многих случаях такие маршруты задаются на этапе начальной загрузки с помощью одного из сценариев запуска системы. Например, команды

```
route add -net 132.236.220.64 netmask 255.255.255.192 gw 132.236.212.6 eth1
route add default gw 132.236.227.1 eth0
```

добавляют, соответственно, четвертый и второй маршруты из числа тех, что отображаются выше командой `netstat -rn` (первый и третий маршруты добавляются командой `ifconfig` при конфигурировании интерфейсов `eth0` и `eth1`).

Последний маршрут также добавляется на этапе начальной загрузки. Он определяет псевдоустройство, называемое интерфейсом обратной связи (loopback interface). Это устройство препятствует пакетам, которые компьютер посылает сам себе, выходить в сеть. Вместо этого они напрямую переносятся из выходной сетевой очереди ядра во входную очередь.

В относительно стабильной локальной сети статическая маршрутизация является достаточно эффективным решением. Эта система проста в управлении и надежна в эксплуатации, но требует знания системным администратором топологии сети на момент начальной загрузки, а также, чтобы эта топология в периоды между загрузками не изменялась.

Большинство компьютеров локальной сети имеет единственный выход во внешний мир, поэтому маршрутизация осуществляется очень просто: достаточно на этапе начальной загрузки добавить стандартный маршрут. Компьютер может получить стандартный маршрут вместе со своим IP-адресом у сервера DHCP (этот протокол описан в разделе 12.9).

В сетях с более сложной топологией требуется динамическая маршрутизация. Она осуществляется процессом-демоном, который ведет и модифицирует таблицу маршрутизации. Демоны маршрутизации, “обитающие” на различных компьютерах, взаимодействуют друг с другом с целью определения топологии сети и решения вопроса о том, как добраться до удаленных адресатов. Существует несколько таких демонов.

Переадресующие ICMP-пакеты

В принципе протокол IP не предусматривает средств управления маршрутной информацией, но в нем имеется механизм контроля нарушений: директивы переадреса-

ции протокола ICMP. Если маршрутизатор направляет пакет компьютеру, находящемуся в той же сети, из которой этот пакет был получен, то что-то работает неправильно. Поскольку отправитель, маршрутизатор и маршрутизатор следующего перехода находятся в одной сети, то пакет можно послать не через два перехода, а через один. Маршрутизатор делает вывод о том, что таблицы маршрутизации отправителя являются неточными или неполными.

В этой ситуации маршрутизатор может уведомить отправителя о проблеме с помощью переадресующего ICMP-пакета. В таком пакете сообщается следующее: "Не нужно посылать мне пакеты для хоста *xxx*; их следует адресовать хосту *yyy*". Протокол ICMP позволяет посылать директивы переадресации как отдельным компьютерам, так и целым сетям. Во многих реализациях, однако, допускается создание директив только первого типа. Директивы второго типа в основном бесполезны, так как применимы лишь к сетям типов A, B и C.

Получив директиву переадресации, отправитель должен обновить свою таблицу маршрутизации, чтобы следующие пакеты, предназначенные данному адресату, шли по более прямому пути. Раньше, когда только появилась технология группового вещания, некоторые системы выдавали переадресующие ICMP-пакеты в ответ на получение групповых пакетов. Современные системы избавлены от подобной проблемы.

Стандартный ICMP-сценарий не подразумевает этапа аутентификации. Предположим, маршрутизатор получает от другого, якобы известного ему маршрутизатора команду перенаправить трафик в другое место. Нужно ли подчиниться этой команде? В действительности такой сценарий создает определенные проблемы с точки зрения безопасности, поэтому переадресующие директивы игнорируются Linux-системами и маршрутизаторами Cisco. Считается, что не стоит доверять непроверенным хостам модификацию таблицы маршрутизации.

В Linux допустимость ICMP-директив определяется элементом `accept_redirects` файловой системы `/proc`. О том, как проверять и устанавливать этот и подобные ему параметры, рассказывается в разделе 12.10.

12.6. ARP: протокол преобразования адресов

Несмотря на то что идентификация IP-пакетов осуществляется при помощи IP-адресов, для фактической передачи данных на канальном уровне должны применяться аппаратные адреса[14]. ARP (Address Resolution Protocol — протокол преобразования адресов) определяет, какой аппаратный адрес связан с тем или иным IP-адресом. Этот протокол можно применять в сетях любых типов, которые поддерживают широковещательный режим, но чаще всего его рассматривают в контексте сетей Ethernet.

Когда компьютер А хочет послать пакет компьютеру Б, находящемуся в том же Ethernet-сегменте, он использует протокол ARP для нахождения аппаратного адреса Б. Если же Б расположен в другой сети, то А с помощью подсистемы маршрутизации определяет IP-адрес маршрутизатора следующего перехода, а затем с помощью протокола ARP выясняет аппаратный адрес этого маршрутизатора. Так как в ARP применяются широковещательные пакеты, которые не могут выйти за пределы локальной сети[15], этот

[14] Это не касается двухточечных соединений, где получатель иногда подразумевается неявно.

[15] Маршрутизатор можно сконфигурировать так, чтобы он пропускал широковещательные пакеты в другие сети. Не делайте этого!

протокол позволяет находить только адреса устройств, непосредственно подключенных к той же сети.

Каждый компьютер хранит в памяти специальную таблицу, называемую *кэшем ARP*. Кэш содержит результаты последних ARP-запросов. В нормальных условиях многие адреса, необходимые компьютеру, выявляются вскоре после начальной загрузки, поэтому протокол ARP мало влияет на загруженность сети.

Протокол ARP функционирует путем широковещательной рассылки[16] пакетов примерно следующего содержания: "Знает ли кто-нибудь аппаратный адрес для 128.138.116.4?" Устройство, которое разыскивают, узнает свой IP-адрес и посылает ответ: "Да, это IP-адрес одного из моих сетевых интерфейсов, соответствующий Ethernet-адрес — 8:0:20:0:fb:6a".

Исходный запрос включает IP-адрес и Ethernet-адрес запрашивающей стороны, благодаря чему разыскиваемое устройство может ответить, не посылая собственный ARP-запрос. Это позволяет обоим компьютерам узнать адреса друг друга за один сеанс обмена пакетами. Другие компьютеры, "слышавшие" исходное широковещательное сообщение, посланное инициатором запроса, тоже могут записать информацию о его адресах. Иногда такую пассивную проверку ARP-трафика называют *слежкой* (snooping).

Команда `arp` изучает и обрабатывает содержимое кэша ARP, добавляет или удаляет из него записи и выполняет запись данных на диск или отображает таблицу. В частности, команда `arp -a` отображает содержимое кэша, например:

```
redhat$ /sbin/arp -a
sprint-gw (192.168.1.254) at 00:02:4B:5B:26:45 [ether] on eth0
inura-local.toadranch.com (192.168.1.101) at 00:04:76:37:AE:7E [ether] on eth0
```

Команда `arp`, как правило, применяется в целях отладки и при работе со специальным оборудованием. Некоторые устройства не могут общаться по протоколу ARP (например, сетевые принтеры и специализированные графические мониторы), и для включения их в сеть нужно сконфигурировать другой компьютер в качестве прокси-сервера ARP. Обычно для этого служит команда `arp -s`:

```
# /sbin/arp -s printer.toadranch.com 00:50:04:ce:ef:38
```

Ядра Linux версии 2.4 не поддерживают прокси-сервер ARP для всей подсети, но автоматически выступают в качестве прокси-сервера, если маршрут существует и интерфейс сконфигурирован на пересылку пакетов.

Если два хоста сети имеют одинаковый IP-адрес, то на одном из них запись в ARP-таблице будет правильной, а на другом — нет. С помощью команды `arp` можно найти хост-нарушитель.

Иногда аппаратные адреса нужно транслировать в IP-адреса. Многие специализированные аппаратные устройства (бездисковые рабочие станции, сетевые компьютеры, принтеры) делают это во время начальной загрузки. Не имея жестко заданного IP-адреса в файле конфигурации, устройство обращается к центральному серверу с просьбой сообщить ему адрес.

На этом мы завершаем рассмотрение основных концепций построения сетей. В следующих разделах рассмотрены вопросы конфигурирования компьютеров, работающих под управлением Linux, для подключения к локальной сети и к Internet.

[16] В ARP применяются средства широковещания канального уровня, а не протокола IP.

12.7. Подключение компьютера к сети

Процесс подключения нового компьютера к существующей локальной сети состоит всего из нескольких этапов. Проблема заключается в том, что в некоторых системах файлы, подлежащие модификации, глубоко спрятаны, поэтому задача усложняется.

В других системах имеется инсталляционный сценарий, отображающий приглашение на ввод соответствующих сетевых параметров. Работать со сценарием удобно до тех пор, пока не потребуется отменить какие-нибудь установки или переместить компьютер в другое место. Прежде чем подключать компьютер к сети, где есть выход в Internet, его нужно оснастить средствами защиты, чтобы ненароком не привлечь в локальную сеть хакеров.

Основные этапы подключения компьютера таковы:

- назначение компьютеру уникального IP-адреса и сетевого имени;
- настройка компьютера на конфигурирование своих сетевых интерфейсов в процессе начальной загрузки;
- задание стандартного маршрута и, возможно, других параметров маршрутизации;
- настройка DNS-сервера, чтобы к компьютеру можно было получать доступ через Internet.

Естественно, к этой последовательности можно добавить этап отладки. После любого изменения, которое способно повлиять на ход начальной загрузки, необходимо обязательно перезагрузиться и проверить правильность функционирования системы. Ведь когда через полгода произойдет внезапный сбой питания и система откажется загружаться, будет очень трудно вспомнить, какие изменения стали источником проблем.

Если в сети используется сервер DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования хостов), он возьмет на себя вышеперечисленные обязанности. Подробнее об этом будет рассказываться в разделе 12.9.

Ниже мы опишем процедуру подключения компьютера в общем виде, а детали, касающиеся конкретных дистрибутивов, будут рассмотрены в разделе 12.8. Соответствующие команды одинаковы во всех дистрибутивах, но у каждого дистрибутива свои конфигурационные файлы, с помощью которых осуществляется автоматическая настройка сети на этапе начальной загрузки (табл. 12.9).

Таблица 12.9. Сетевые конфигурационные файлы

Система	Файл	Содержимое
Red Hat, Fedora	`/etc/sysconfig/network`	Имя компьютера, стандарт-ный маршрут
	`network-scripts/ifcfg-`*`интерфейс`*	IP-адрес, сетевая маска, широковещательный адрес
SUSE	`/etc/rc.config`	Имя компьютера, IP-адрес, сетевая маска и др.
	`/etc/route.conf`	Стандартный маршрут
Debian, Ubuntu	`/etc/hostname`	Имя компьютера
	`/etc/network/interfaces`	IP-адрес, сетевая маска, стандартный маршрут

Процесс проектирования физической сети описан в главе 14. Если вы имеете дело с существующей сетью и в общих чертах знаете, как она организована, то, наверное, чи-

тать о физических аспектах сетей имеет смысл только в том случае, когда планируется расширение действующей сети.

Мы рассмотрим процесс конфигурирования компьютера в сети Ethernet. Для других топологий процедура мало чем отличается.

Присвоение сетевых имен и IP-адресов

📖 Подробнее о DNS рассказывается в главе 15.

Существует множество теорий о том, как лучше всего определять соответствия между именами компьютеров и IP-адресами в локальной сети: с помощью файла `hosts`, СУБД NIS, службы DNS или комбинации этих средств. Если используется несколько систем, необходимо продумать их совместную работу. Конечная система должна учитывать возможность наращивания сети, легко конфигурироваться и позволять компьютеру загружаться даже в тех случаях, когда не все службы доступны (а также уметь работать в гетерогенной среде).

Другой важный аспект, который следует учитывать, связан со сменой адресов. При смене провайдера может потребоваться вернуть старые IP-адреса и назначить компьютерам адреса, взятые у нового провайдера. Это довольно неприятная процедура, если администратор должен лично посетить каждый компьютер и вручную сконфигурировать его. Чтобы не усложнять себе жизнь, указывайте в конфигурационных файлах сетевые имена, а привязки к IP-адресам храните только в DNS. С другой стороны, наличие IP-адресов в конфигурационных файлах упрощает загрузку компьютера в случае, когда не все службы доступны. Такова судьба системных администраторов: из двух зол приходится выбирать наименьшую.

Использование файла `/etc/hosts` — старейший и простейший способ преобразования сетевых имен в IP-адреса. Каждая строка файла начинается с IP-адреса и содержит различные символьные имена, под которыми известен адрес.

Ниже показан пример содержимого файла `/etc/hosts` для хоста `lollipop`:

```
127.0.0.1           localhost
192.108.21.48       lollipop.xor.com lollipop loghost
192.108.21.254      chimchim-gw.xor.com chimchim-gw
192.108.21.1        ns.xor.com ns
192.225.33.5        licenses.xor.com license-server
```

Минимальная версия файла должна содержать первые две строчки. Чаще всего в первой записи файла определяется хост `localhost`.

Поскольку файл `/etc/hosts` содержит лишь локальные адреса привязки, в большинстве современных систем он используется для хранения адресов, требующихся при начальной загрузке. Остальные локальные и глобальные адреса проверяются в базе данных DNS. Иногда в файле `/etc/hosts` хранятся записи, о которых не следует “знать” другим компьютерам и которых нет в DNS.

Файл `/etc/hosts` когда-то был важен для начальной загрузки, так как сервер DNS еще не был доступен, а в конфигурационных файлах иногда содержались сетевые имена, а не IP-адреса. Современным дистрибутивам Linux файл `/etc/hosts` не очень-то нужен, но он должен содержать хотя бы адреса самого компьютера и адрес интерфейса обратной связи. Кроме того, может оказаться полезным наличие адреса стандартного шлюза и адреса сервера имен. Во многих системах в этот файл помещаются адреса всех важных хостов, серверов и шлюзов, иногда — адреса всех локальных хостов и внешних резервных серверов имен.

В наших демонстрационных дистрибутивах имеется лишь самая простая версия файла `/etc/hosts`. В Debian и Ubuntu в этом файле находится запись хоста `localhost` адреса 127.0.0.1, сетевое имя системы, установленной по адресу 127.0.1.1 и определенная информация IPv6. В Red Hat и Fedora — это записи хоста `localhost` и самого компьютера, в SUSE — то же, что и в Red Hat, плюс адреса IPv6 хоста `localhost` и еще несколько специальных имен IPv6.

Команда `hostname` назначает компьютеру сетевое имя. Она обычно вызывается на этапе начальной загрузки из сценария запуска системы, который запрашивает назначаемое имя из конфигурационного файла. Естественно, все поставщики систем называют этот файл по-разному (информация о конкретных дистрибутивах приведена в разделе 12.8). В большинстве современных систем компьютеру назначается полностью определенное доменное имя (т.е. оно включает как имя хоста, так и имя домена DNS, например `anchor.cs.colorado.edu`).

🕮 Дополнительная информация о протоколе LDAP приведена в разделе 17.5.

В небольшой организации вполне можно выделять IP-адреса и сетевые имена вручную. Когда же в организации множество сетей и разнородных административных групп, лучше придерживаться принципа централизации. Об уникальности динамически назначаемых сетевых параметров заботится сервер DHCP. В некоторых организациях сетевыми именами и IP-адресами управляют с помощью баз данных LDAP.

Команда `ifconfig`: конфигурирование сетевых интерфейсов

Команда `ifconfig` используется для подключения и отключения сетевого интерфейса, а также задания его IP-адреса, маски подсети, других опций и параметров. Она обычно выполняется на этапе начальной загрузки (аргументы командной строки берутся из конфигурационного файла), но может применяться и для внесения изменений в работающую систему. Будьте осторожны при модификации удаленной системы, так как не всегда есть возможность оперативно исправить ошибки.

В большинстве случаев команда `ifconfig` имеет следующий формат:

```
ifconfig интерфейс адрес опции ...
```

Например:

```
ifconfig eth0 192.168.1.13 netmask 255.255.255.0 up
```

Параметр *интерфейс* обозначает аппаратный интерфейс, к которому применяется команда. В UNIX-системах это обычно двух- или трехсимвольное имя устройства (определяемое комплектом микросхем сетевой платы), за которым следует число, но в Linux это почти всегда имя вида `eth0`[17]. Реальный идентификатор устройства и ссылка на соответствующий драйвер хранятся в файле `/etc/modprobe.conf` в строке `alias`. Имя интерфейса обратной связи — `lo`.

[17] Одному интерфейсу можно назначить несколько IP-адресов, воспользовавшись концепцией “виртуальных” сетевых интерфейсов, или “IP-псевдонимов”. Администраторы часто прибегают к этому, когда на одном компьютере требуется разместить несколько Web-сайтов. В Linux виртуальные интерфейсы называются eth0:0, eth0:1 и т.д. Заранее объявлять о наличии этих интерфейсов не нужно. Команда `ifconfig` автоматически активизирует их, получив соответствующие указания. Подробнее об этом рассказывается в разделе 21.3.

Команда `ifconfig` *интерфейс* отображает текущие установки для указанного интерфейса. Во многих системах опция `-a` понимается как "все интерфейсы", поэтому с помощью команды `ifconfig -a` можно узнать, какие интерфейсы присутствуют в системе. Если эта команда не поддерживается, попробуйте выполнить команду `netstat -i`.

Параметр *адрес* задает IP-адрес интерфейса. Многие версии команды `ifconfig` принимают также имя хоста в качестве адресного параметра. Мы предпочитаем пользоваться числовой записью. Если команде `ifconfig` будет передано имя хоста (или результат работы команды `hostname`) и в процессе преобразования имени возникнет ошибка, то компьютер не загрузится или загрузится в состояние, при котором доступ к нему по сети будет невозможен. Администратору придется устранять проблему на месте. DNS-запросы в подобной ситуации выполняются очень долго, поэтому возникает впечатление, будто система зависла. С другой стороны, тот, кому доводилось менять адреса компьютеров во всей сети, знает, что поиск всех этих жестко заданных IP-адресов в конфигурационных файлах иногда оборачивается настоящим кошмаром.

Ключевое слово `up` указывает на активизацию интерфейса, а ключевое слово `down` — на его отключение. Когда команда `ifconfig` назначает интерфейсу IP-адрес, ключевое слово `up` подразумевается неявно и может быть опущено.

Команда `ifconfig` понимает множество опций. Мы рассмотрим лишь самые основные из них; как всегда, детали, касающиеся конкретной системы, можно узнать на страницах интерактивного руководства. Все опции имеют символические имена. Некоторые опции требуют наличия аргументов, которые должны следовать сразу за названием опции.

Опция `netmask` задает маску подсети для данного интерфейса. Эта опция обязательна, если подсеть формируется не на основании класса адреса (A, B или C). Маску можно указывать в точечной нотации либо в виде четырехбайтового шестнадцатеричного числа, начинающегося с префикса `0x`. В любом случае единичные биты являются частью номера сети, а нулевые биты — частью номера хоста.

Опция `broadcast` задает широковещательный IP-адрес интерфейса в шестнадцатеричной или точечной записи. Правильный широковещательный адрес — тот, в котором все биты машинной части равны единице. В большинстве систем по умолчанию используется именно такой адрес. Он определяется на основании сетевой маски и IP-адреса.

В Linux в качестве широковещательного адреса можно использовать любой IP-адрес, допустимый в сети, к которой подключен компьютер. В некоторых организациях широковещательный адрес специально выбран нестандартным для предотвращения сетевых атак, основанных на широковещательной команде `ping`. По ряду причин мы не рекомендуем такой подход.

Во-первых, для этого требуется сбросить широковещательные адреса на всех компьютерах локальной сети, что весьма накладно в крупной организации. Во-вторых, необходимо быть абсолютно уверенным в том, что переконфигурации подверглись все компьютеры, иначе возникнет лавина широковещательных запросов, т.е. пакеты будут переходить от хоста к хосту до тех пор, пока не истечет время их жизни (параметр TTL станет равным нулю).

Возникновение широковещательного "шторма" объясняется тем, что для передачи пакетов применяется один и тот же широковещательный адрес канального уровня независимо от того, каким является широковещательный IP-адрес. Предположим, к примеру, что для компьютера X широковещательный адрес равен A1, а для компьютера Y — A2. Если X посылает пакет по адресу A1, то Y примет этот пакет (его целевой адрес канального уровня является широковещательным), обнаружит, что он не предназначен

ему и не является широковещательным (ведь для Y таковым служит адрес A2), и вернет[18] его обратно в сеть. Если таких компьютеров, как Y, два, пакет будет циркулировать до тех пор, пока время его жизни не истечет. Широковещательный "шторм" приводит к серьезному снижению производительности крупных коммутируемых сетей.

Лучший способ избежать проблем с широковещательными запросами — запретить пограничному маршрутизатору перенаправлять их, а отдельным хостам — отвечать на них. В разделе 12.10 будет рассказано о том, как реализовать подобные ограничения.

В показанном в начале подраздела примере широковещательный адрес равен 192.168.1.255, так как длина сетевой части адреса — 24, что следует из значения сетевой маски (255.255.255.0).

При выполнении команды `ifconfig eth0` будет получен такой результат:

```
redhat$ /sbin/ifconfig eth0
eth0 Link encap:Ethernet  HWaddr 00:02:B3:19:C8:86
     inet addr:192.168.1.13  Bcast:192.168.1.255  Mask:255.255.255.0
     UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
     RX packets:206983 errors:0 dropped:0 overruns:0 frame:0
     TX packets:218292 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:100
     Interrupt:7 Base address:0xef00
```

Отсутствие коллизий в данном примере может указывать на очень слабо загруженную сеть или, что более вероятно, на коммутируемую сеть. В сети общего доступа (организованной с помощью концентраторов, а не коммутаторов) этот показатель должен быть ниже уровня 5% от числа отправленных пакетов. Большое число коллизий свидетельствует о загруженности сети, которую, возможно, требуется разбить на подсети или перевести на коммутируемую инфраструктуру.

Рассмотрим несколько примеров.

```
# ifconfig lo 127.0.0.1 up
```

Эта команда конфигурирует интерфейс обратной связи, который, как правило, не требует никаких опций. Никогда не следует менять стандартную системную конфигурацию данного интерфейса. Подразумеваемая маска 255.0.0.0 является правильной и не должна переопределяться вручную.

```
# ifconfig eth0 128.138.243.151 netmask 255.255.255.192
        broadcast 128.138.243.191 up
```

Это типичный пример для платы Ethernet. IP-адрес и широковещательный адрес устанавливаются равными 128.138.243.151 и 128.138.243.191, соответственно. Сеть относится к классу B (это видно из первого байта адреса), но вместе с тем в ней выделяется подсеть /26. Значение 192 в сетевой маске — это 11000000 в двоичном виде, т.е. к трем октетам 255 добавляются еще два бита. Значение 191 в широковещательном адресе — это 10111111 в двоичном виде, т.е. все шесть битов машинной части равны 1, а сам интерфейс принадлежит к третьей (10) из четырех подсетей. (Это как раз тот случай, когда может пригодиться программа вычисления IP-адресов!)

Теперь, когда мы рассмотрели процедуру ручного конфигурирования сетевого интерфейса, осталось выяснить, как задавать параметры команды `ifconfig` на этапе начальной загрузки системы. Кроме того, необходимо убедиться в том, что новые значения введены правильно. Для этого потребуется отредактировать один или несколько конфигурационных файлов. Эта процедура будет описана в разделе 12.8.

[18] Чтобы это произошло, на компьютере Y должна быть включена опция `ip_forwarding`.

Команда `mii-tool`: настройка автоматического согласования и других параметров физической среды

Иногда у сетевого устройства есть настраиваемые параметры, зависящие от типа передающей среды. Чрезвычайно распространенный пример — современные платы Ethernet, которые могут поддерживать скорость передачи 10, 100 и даже 1000 Мбит/с как в полудуплексном, так и в дуплексном режимах. Большинство устройств по умолчанию находится в режиме автоматического согласования, когда обе стороны — сетевая плата и вышестоящее устройство (обычно порт коммутатора) — пытаются угадать параметры друг друга.

К сожалению, процедура автоматического согласования больше напоминает попытку вставить ключ в замочную скважину с завязанными глазами. Современные сетевые устройства лучше взаимодействуют друг с другом, но автоматическое согласование все еще часто приводит к сбоям в работе оборудования. Следствием неудачного согласования является высокий коэффициент потери пакетов (особенно больших пакетов).

Наилучший способ избежать подобной ловушки — зафиксировать скорости и режимы работы сетевых интерфейсов как на серверах, так и на портах коммутатора, к которым они подключены. Автоматическое согласование удобно в динамических сетях организаций, где многие сотрудники имеют портативные компьютеры, и совершенно не нужно для статически подключенных компьютеров. Если в сети начинают происходить загадочные потери пакетов, в первую очередь отключите везде режим автоматического согласования.

В Linux существует команда `mii-tool`, которая запрашивает и устанавливает физические параметры устройств, такие как скорость передачи по каналу и дуплексный режим. Для определения параметров сетевого интерфейса предназначен флаг `-v`. Например, следующая команда сообщает о том, что интерфейс `eth0` находится в режиме автоматического согласования:

```
$ mii-tool -v eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
  product info: vendor 00:10:5a, model 0 rev 0
  basic mode:   autonegotiation enabled
  basic status: autonegotiation complete, link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
  link partner: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
```

Чтобы зафиксировать скорость работы 100 Мбит/с и дуплексный режим, воспользуйтесь такой командой:

```
# mii-tool -force=100BaseTx-FD eth0
```

Чтобы она выполнялась всегда, добавьте ее в сценарий загрузки системы. После этого запрос о состоянии системы буде возвращать следующую информацию:

```
$ mii-tool -v eth0
eth0: 100 Mbit, full duplex, link ok
  product info: vendor 00:10:5a, model 0 rev 0
  basic mode:   100 Mbit, full duplex
  basic status: link ok
  capabilities: 100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD
  advertising:  100baseTx-FD 100baseTx-HD 10baseT-FD 10baseT-HD flow-control
```

Команда route: конфигурирование статических маршрутов

Команда route определяет статические маршруты — явно заданные элементы таблицы маршрутизации, которые обычно не меняются даже в тех случаях, когда запускается демон маршрутизации. При подключении нового компьютера к локальной сети достаточно указать стандартный маршрут (об этом читайте в следующем подразделе).

Маршрутизация рассматривается нами в двух главах: этой и следующей. И хотя информация по основам маршрутизации и команде route приведена здесь, полезно прочитать также несколько первых разделов главы 14.

Маршрутизация осуществляется на сетевом уровне. Когда приходит пакет, предназначенный для другого хоста, его целевой IP-адрес сравнивается с записями в таблице маршрутизации ядра. При совпадении (хотя бы частичном) с каким-нибудь маршрутом в таблице пакет направляется по IP-адресу следующего шлюза, связанного с данным маршрутом.

Но есть два особых случая. Во-первых, пакет может быть адресован компьютеру, включенному в ту же сеть, что и хост-отправитель. В этом случае адрес следующего шлюза соответствует одному из интерфейсов локального компьютера, и пакет посылается прямо в пункт назначения. Маршруты такого типа добавляются командой ifconfig при конфигурировании интерфейса.

Во-вторых, может вообще не оказаться маршрута, совпадающего с адресом пункта назначения. В этом случае применяется маршрут по умолчанию, если таковой имеется, иначе отправителю посылается ICMP-сообщение “network unreachable” (сеть недоступна) или “host unreachable” (хост недоступен). Многие локальные сети имеют единственный выход во внешний мир, на него-то и ссылается стандартный маршрут. В магистральной сети Internet нет стандартных маршрутов — это конец пути.

Каждая команда route добавляет или удаляет один маршрут. Ее формат таков:

route [*операция*] [*тип*] *адресат* **gw** *шлюз* [*метрика*] [**dev** *интерфейс*]

Параметр *операция* принимает одно из двух значений: add (добавить маршрут) или del (удалить маршрут). При отображении содержимого таблицы этот аргумент опускают. Параметр *адресат* содержит адрес хоста (опция -host), адрес сети (опция -net) или ключевое слово default. Если указан адрес сети, то должна быть задана и сетевая маска.

Параметр *шлюз* определяет устройство, которому будут пересылаться пакеты. Это устройство должно находиться в той же сети, что и хост, на котором формируется маршрут. Пересылка выполняется только по одному переходу за раз. В Linux вместо шлюза (или наряду с ним) может указываться интерфейс. Ключевое слово dev в спецификации интерфейса является необязательным.

Параметр *метрика* задает предельное число переходов на пути к адресату. В Linux этот параметр необязателен, но если он указан, то значение сохраняется в маршрутных таблицах, чтобы им могли воспользоваться демоны маршрутизации.

Необязательный параметр *тип* задает тип маршрута: сетевой или хостовой. Соответственно, он может принимать значение -net или -host. Во втором случае целевой IP-адрес соответствует хосту, а не сети. Если тип опущен, команда route проверяет машинную часть целевого адреса на предмет равенства нулю. Когда машинная часть

равна нулю или адрес найден в файле /etc/networks, маршрут считается обычным сетевым маршрутом[19].

Поскольку команда route не "знает", разбита ли сеть с указанным адресом на подсети, тип адреса приходится задавать достаточно часто. Например, адрес 128.138.243.0 относится к сети класса B, имеющейся в нашей организации и разделенной на подсети, но для команды route он выглядит как адрес класса B 128.138 с машинной частью 243.0. Чтобы не вводить команду route в заблуждение, следует воспользоваться опцией -net. Желательно указывать тип адреса для всех маршрутов, где могут встретиться подсети.

Команда route del *адрес* удаляет соответствующую запись из таблицы маршрутизации. В других UNIX-системах у команды route есть специальная опция, обычно -f или -flush, при наличии которой команда удаляет из таблицы данные обо всех маршрутах. В Linux такая опция не поддерживается, поэтому при очистке большой таблицы придется вводить много команд route del. Это лучше делать, будучи зарегистрированным локально, иначе внезапный разрыв связи приведет к тому, что таблица останется в несогласованном состоянии.

Существующие маршруты можно проанализировать с помощью команды netstat -nr или netstat -r (в этом случае вместо числовых адресов будут отображаться сетевые имена). Первый вариант лучше при отладке, поскольку неисправность может заключаться как раз в модуле преобразования имен.

```
redhat$ netstat -nr
Kernel IP routing table
Destination     Gateway         Genmask         Flags  MSS  Window  irtt  Iface
192.168.1.0     0.0.0.0         255.255.255.0   U      0    0       0     eth0
127.0.0.0       0.0.0.0         255.0.0.0       U      0    0       0     lo
0.0.0.0         192.168.1.254   0.0.0.0         UG     0    0       0     eth0
redhat$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags  MSS  Window  irtt  Iface
192.168.1.0     *               255.255.255.0   U      0    0       0     eth0
127.0.0.0       *               255.0.0.0       U      0    0       0     lo
default         sprint-gw       0.0.0.0         UG     0    0       0     eth0
```

В колонке Genmask указана маска целевого адреса. Флаги поля Flags указывают на статус маршрута, на то, как он был получен, и т.д. Наконец, в колонке Iface указан интерфейс, через который должны направляться пакеты, следующие по данному маршруту. Эти примеры взяты из Red Hat, но в SUSE и Debian результаты будут аналогичными, за исключением того, что в Debian маршрут через интерфейс обратной связи не отображается по умолчанию.

Стандартные маршруты

Благодаря наличию стандартного маршрута все пакеты, целевой адрес которых не найден в таблице маршрутизации ядра, будут направляться в указанный шлюз. Для создания стандартного маршрута достаточно добавить в один из сценариев запуска системы такую строку:

```
route add default gw IP-адрес
```

[19] Файл /etc/networks используется для привязки доменных имен к сетевым адресам подобно тому, как файл /etc/hosts связывает имена компьютеров с их IP-адресами. Многие команды, требующие указания сетевых адресов, понимают также доменные имена, если они упомянуты в файле /etc/networks (или в базе данных DNS).

Чтобы не задавать явный IP-адрес в сценариях запуска, большинство поставщиков заставили свои системы читать адрес из конфигурационного файла. К сожалению, в разных дистрибутивах эти процедуры различны. Это явный признак известного синдрома "придумано не мной", и остается надеяться, что участники проекта LSB21 не оставят сей факт без внимания. В табл. 12.10 сведена информация по нашим демонстрационным дистрибутивам.

Таблица 12.10. Задание стандартного маршрута в различных дистрибутивах Linux

Система	Редактируемый файл	Изменяемая переменная
Red Hat, Fedora	`/etc/sysconfig/network`	`GATEWAY`
SUSE	`/etc/route.conf`	добавить строку: `default` *IP-адрес маска интерфейс*
Debian, Ubuntu	`/etc/network/interfaces`	`gateway`

Конфигурирование DNS

Чтобы сконфигурировать компьютер в качестве DNS-клиента, достаточно отредактировать один или два файла: `/etc/resolv.conf` (во всех системах) и файл "переключения служб" (в некоторых системах).

Файл `/etc/resolv.conf` содержит список DNS-доменов, просматриваемых при анализе неполных имен (например, "anchor" вместо anchor.cs.colorado.edu), и список IP-адресов серверов имен, в которых осуществляется поиск имен. Ниже показан пример этого файла; подробнее о нем рассказывается в разделе 15.10.

```
search cs.colorado.edu colorado.edu
nameserver 128.138.242.1
nameserver 128.138.243.151
nameserver 192.108.21.1
```

Первым должен быть приведен "ближайший" стабильный сервер имен, так как он опрашивается в первую очередь. Всего можно задать три записи `nameserver`. Желательно указывать более одного сервера. Период тайм-аута DNS-запроса к конкретному серверу имен достаточно велик, поэтому, если первый сервер не отвечает, пользователи это заметят.

Иногда вместо записи `search` присутствует запись `domain`. Это свидетельствует о том, что в системе либо старая версия файла `resolv.conf`, либо старый модуль преобразования имен, который не понимает директиву `search`. Директива `domain` определяет только текущий домен, а директива `search` позволяет включать в запрос до шести различных имен доменов и поэтому является более предпочтительной.

📖 Информация о NIS приведена в главе 17.

Кроме того, в некоторых старых системах служба DNS по умолчанию не используется, даже если имеется правильно сконфигурированный файл `resolv.conf`. В таких системах существует файл "переключения служб", определяющий, какие механизмы должны применяться для преобразования сетевых имен в IP-адреса. О приоритетах информационных служб подробно рассказывается в разделе 17.4, но здесь мы также рассмотрим эту тему, поскольку эта информация необходима для конфигурирования старого компьютера.

Файл "переключения служб" позволяет задать порядок просмотра баз данных, таких как DNS и NIS, и файла `/etc/hosts`. Некоторые информационные источники можно полностью исключить из рассмотрения. Порядок просмотра оказывает влияние на начальную загрузку системы и обработку содержимого файла `/etc/hosts`.

Если первым источником является DNS, то в локальной сети должен присутствовать сервер имен, а его сетевое имя и IP-адрес должны быть указаны в файле `/etc/hosts`, иначе могут возникнуть проблемы при загрузке.

В табл. 12.11 указано местонахождение соответствующих конфигурационных файлов в каждом из наших дистрибутивов и принятые в них стандартные установки.

Таблица 12.11. Файлы "переключения служб" в различных дистрибутивах

Система	Файл	Приоритет источников информации
Ubuntu	`/etc/nsswitch.conf`	`files dns mdns`[a]
	`/etc/host.conf`	`hosts, bind`
Остальные системы	`/etc/nsswitch.conf`	`files dns`
	`/etc/host.conf`	`hosts, bind`

[a] mdns = multicast DNS (групповая DNS-адресация), несколько экзотический протокол, который позволяет выполнять преобразование DNS-подобных имен в небольшой сети, не имеющей DNS-сервера.

Стек сетевых протоколов Linux

📖 Подробнее об этом рассказывается в разделе 21.3.

Стек сетевых протоколов в ядрах Linux версии 2.2 и выше включает средства поддержки виртуальных сетевых интерфейсов, избирательных подтверждений (selective acknowledgement, SACK). В ядрах версии 2.4 и последующих реализованы явные уведомления о перегрузке (Explicit Congestion Notification, ECN).

Чтобы информировать удаленный компьютер о наличии проблем перегрузки, ECN помечает пакеты TCP, а не просто передает информацию о пропущенных пакетах. Впервые механизм ECN был описан в документе RFC2481 (январь 1999 г.), и теперь он предложен в качестве стандарта (RFC3168). В документе RFC2884 (июль 2000 г.) оценивается производительность нового механизма. Из него следует, что новый механизм предпочтителен для множества сетевых транзакций.

Стек сетевых протоколов Linux всегда быстро вбирает в себя все новое. Впрочем, иногда разработчики оказываются "впереди планеты всей". Например, вскоре после внедрения механизма ECN (который по умолчанию был активизирован) выяснилось, что он конфликтует с неправильными стандартными установками старых версий брандмауэров Cisco, вследствие чего все пакеты с установленным битом ECN удалялись. Неприятный сюрприз!

Разработчики Linux любят экспериментировать и часто реализуют функции и алгоритмы, которые еще не приняты в качестве стандарта. Примером могут служить добавленные в Linux 2.6.13 подключаемые алгоритмы управления перегрузкой. Они включают в себя различные параметры для сетей, в которых происходят потери пакетов, для высокоскоростных глобальных сетей с множеством пропускаемых пакетов, спутниковых линий связи и многого другого. Стандартный механизм "восстановления" TCP (медленный запуск, предотвращение перегрузки, быстрая повторная передача и быстрое восста-

новление данных) все еще используется по умолчанию, но новый вариант может оказаться более эффективным.

12.8. Сетевое конфигурирование в различных дистрибутивах

В главе 2 были подробно описаны процедуры начальной загрузки для всех наших дистрибутивов. В следующих подразделах мы просто подытожим все, что связано с конфигурированием сети. Интерфейс обратной связи всегда конфигурируется автоматически; эту часть процесса модифицировать не нужно. В остальном же все дистрибутивы несколько отличаются друг от друга.

Четыре файла являются общими для всех наших дистрибутивов: `/etc/hosts`, `/etc/resolv.conf`, `/etc/nsswitch.conf` и `/etc/host.conf`. Все они были описаны выше, в разделе общего сетевого конфигурирования. За исключением файла `resolv.conf` и, возможно, `hosts`, остальные файлы не должны модифицироваться при подключении нового компьютера к сети.

После внесения изменений в файл, от которого зависит сетевое конфигурирование на этапе начальной загрузки, необходимо либо перезагрузить систему, либо отключить сетевой интерфейс, а затем снова его подключить, чтобы изменения вступили в силу. Во всех наших дистрибутивах можно использовать команды `ifup` и `ifdown`.

Сетевое конфигурирование в Red Hat и Fedora

В табл. 12.12 перечислены конфигурационные файлы Red Hat и Fedora, в которых содержатся параметры сети.

Таблица 12.12. Сетевые конфигурационные файлы Red Hat и Fedora

Файл в каталоге `/etc/sysconfig`	Содержимое
`network`	Имя компьютера, стандартный маршрут
`static-routes`	Статические маршруты
`network-scripts/ifcfg-`*`интерфейс`*	Параметры конкретного интер-фейса: IP-адрес, сетевая маска и т.п.

Сетевое имя компьютера задается в файле `/etc/sysconfig/network`, где также определяются DNS-домен и стандартный шлюз. К примеру, ниже показано содержимое файла `network` для компьютера, в котором установлена одна Ethernet-плата:

```
NETWORKING=yes
HOSTNAME=redhat.toadranch.com
DOMAINNAME=toadranch.com        ### необязательное поле
GATEWAY=192.168.1.254
```

Данные, касающиеся конкретных интерфейсов, находятся в файлах `/etc/sysconfig/network-scripts/ifcfg-`*`интерфейс`*, где последний компонент представляет собой имя сетевого интерфейса. В этих файлах задаются IP-адрес, сетевая маска, сетевой и широковещательный адреса каждого интерфейса. Имеется также строка, в которой можно указать, должен ли интерфейс активизироваться на этапе начальной загрузки.

Обычно в системе присутствуют файлы для Ethernet-платы (eth0) и интерфейса обратной связи (lo). Например, записи

```
DEVICE=eth0
IPADDR=192.168.1.13
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
```

и

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
ONBOOT=yes
NAME=loopback
```

представляют собой содержимое файлов ifcfg-eth0 и ifcfg-lo компьютера redhat.toadranch.com, который был описан в файле конфигурирования сети в начале этого раздела.

В Red Hat есть ряд удобных сценариев, которые упрощают управление сетевыми интерфейсами. В частности, сценарии ifup и ifdown принимают в качестве аргумента имя интерфейса и, соответственно, подключают или отключают интерфейс. После изменения файла в каталоге /etc/sysconfig не забудьте выполнить последовательно команды ifdown *интерфейс* и ifup *интерфейс*. А еще лучше — перезагрузить систему, чтобы убедиться в отсутствии скрытых побочных эффектов. Для сценариев ifup и ifdown нет man-страниц, но это обычные сценарии интерпретатора команд (находятся в каталоге /sbin), поэтому достаточно легко определить, что они делают.

Если нужно управлять всеми интерфейсами одновременно, воспользуйтесь сценарием /etc/rc.d/init.d/network, который принимает аргументы start, stop, restart и status. На этапе начальной загрузки этот сценарий вызывается с аргументом start.

Сценарии запуска также могут конфигурировать статические маршруты. Любой маршрут, внесенный в файл /etc/sysconfig/static-routes, добавляется в таблицу маршрутизации на этапе начальной загрузки. Записи этого файла содержат аргументы для команды route add, хотя и в смешанном порядке (интерфейс указан первым, а не последним):

```
eth0 net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49
eth1 net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129
```

После имени интерфейса следуют аргументы команды route: тип маршрута (net или host), адресуемая сеть, маска этой сети и, наконец, шлюз для следующего перехода. Ключевое слово gw необходимо. В современных ядрах Linux не используется параметр metric команды route, но его можно добавить в таблицу маршрутизации, чтобы о нем узнали демоны маршрутизации. Показанные выше записи файла static-routes приведут к формированию следующих команд:

```
route add -net 130.225.204.48 netmask 255.255.255.248 gw 130.225.204.49 eth0
route add -net 192.38.8.0 netmask 255.255.255.224 gw 192.38.8.129 eth1
```

Сетевое конфигурирование в SUSE

В табл. 12.13 приведены файлы конфигурирования сети, используемые в SUSE. SUSE обладает уникальной схемой конфигурирования сети. За исключением DNS-параметров и сетевого имени системы большинство сетевых параметров SUSE сосредоточено в основном в файлах `ifcfg-`*`interface`* в каталоге `/etc/sysconfig/network`. Каждому интерфейсу системы может соответствовать отдельный файл.

Таблица 12.13. Сетевые конфигурационные файлы SUSE, хранящиеся в каталоге `/etc/sysconfig/network`

Файл	Содержимое
`ifcfg-`*`интерфейс`*	Имя компьютера, IP-адрес, сетевая маска и др.
`ifroute-`*`интерфейс`*	Определения маршрутов конкретного интерфейса
`routes`	Стандартный и статические маршруты, используемые всеми интерфейсами
`config`	Множество реже используемых сетевых переменных

Имя файла реальных сетевых интерфейсов (т.е. тех, которые не являются интерфейсами обратной связи) имеет расширенную форму `ifcfg-`*`интерфейс`*`-id-`*`MAC`*, где *`MAC`* — аппаратный адрес сетевого интерфейса (например, `ifcfg-eth-id-00:0c:29:d4:ea:26`).

Кроме IP-адреса, шлюза и информации о вещательных адресах данного интерфейса файлы `ifcfg-*` могут содержать множество других сетевых параметров. Много полезной информации о возможных параметрах можно извлечь из файла шаблона `ifcfg.template`, снабженного подробными комментариями.

Входящая в состав SUSE утилита YaST предоставляет достаточно удобный интерфейс конфигурирования сети. Для управления файлами `ifcfg-*` мы рекомендуем по возможности использовать именно ее. На тот случай, если требуется сконфигурировать сеть вручную, ниже приведен простой шаблон с нашими комментариями:

```
BOOTPROTO='static'       # Статическая маршрутизация подразумевается по
                         # умолчанию, но лишнее определение не повредит.
IPADDR='192.168.1.4/24'  # /24 определяет переменные NETWORK и NETMASK vars
NAME='AMD PCnet - Fast 79C971' # Используется для запуска и остановки интерфейса.
STARTMODE='auto'         # Вызывает автоматический запуск во время загрузки
USERCONTROL='no'         # Отключает управление посредством графического
                         # интерйеса kinternet/cinternet
```

В SUSE информация о глобальных статических маршрутах (в том числе о стандартном маршруте) хранится в файле `routes`. Каждая строка этого файла подобна команде `route` с опущенным именем команды: целевой адрес, шлюз, сетевая маска и необязательные параметры, сохраняемые в таблице маршрутизации для демонов. Например, для сконфигурированного выше хоста, у которого только один стандартный маршрут, содержимое файла будет таким:

```
default 192.168.1.1 - -
```

Уникальные маршруты конкретных интерфейсов хранятся в файлах `ifroute-`*`интерфейс`*, в которых номенклатура компонента *`интерфейс`* соответствует файлам `ifcfg-*`. Формат их содержимого аналогичен формату файла `routes`.

Сетевое конфигурирование в Debian и Ubuntu

Как показано в табл. 12.14, сетевые параметры Debian и Ubuntu сосредоточены в основном в файлах `/etc/hostname` и `/etc/network/interfaces`. Файл `/etc/network/options` играет вспомогательную роль.

Таблица 12.14. Сетевые конфигурационные файлы Debian и Ubuntu

Файл	Содержимое
`/etc/hostname`	Имя компьютера
`/etc/network/interfaces`	IP-адрес, сетевая маска, стандарт-ный маршрут
`/etc/network/options`	Низкоуровневые сетевые опции (перенаправление IP-пакетов и др.)

Сетевое имя компьютера задается в файле `/etc/hostname`. Указанное имя должно быть полностью определенным, поскольку оно часто используется там, где ожидается полное имя. В то же время в стандартной инсталляции Debian в файл заносится краткое имя.

IP-адрес, сетевая маска и стандартный шлюз задаются в файле `/etc/network/interfaces`. Базовый формат записей файла `interfaces` таков: для каждого интерфейса указывается строка, начинающаяся с ключевого слова `iface`, а за ней следуют строки с отступом, в которых указаны дополнительные параметры. Например:

```
iface lo inet loopback
iface eth0 inet static
    address 192.168.1.102
    netmask 255.255.255.0
    gateway 192.168.1.254
```

Команды `ifup` и `ifdown` читают этот файл и, соответственно, подключают или отключают интерфейсы, вызывая низкоуровневые команды, например `ifconfig`, с правильными параметрами. Ключевое слово `inet` в строке `iface` определяет семейство адресов (оно всегда равно `inet`). Ключевое слово `static` обозначает способ описания интерфейса и указывает на то, что IP-адрес и сетевая маска интерфейса `eth0` будут заданы непосредственно. Для статических интерфейсов строки `address` и `netmask` являются обязательными. В первых версиях ядра Linux необходимо было указывать также адрес сети, но современные ядра “умнее” и могут самостоятельно определить его на основании IP-адреса и маски. В строке `gateway` определяется адрес стандартного шлюза.

Файл `options` содержит различные сетевые параметры, устанавливаемые на этапе начальной загрузки. По умолчанию в Debian перенаправление IP-пакетов отключено, защита от поддельных IP-адресов включена, а синхронные файлы “cookie” отключены.

12.9. DHCP: протокол динамического конфигурирования хостов

📖 Протокол DHCP определен в документах RFC2131 и 2132.

Исторически так сложилось, что для добавления Linux-хостов в сеть всегда требовалась ручная настройка. Но если компьютеры Macintosh или Windows без труда подключаются к сети, то почему нельзя делать то же самое в Linux? DHCP (Dynamic Host

Configuration Protocol — протокол динамического конфигурирования хостов) предназначен для решения этой проблемы.

Протокол DHCP дает возможность клиенту взять сетевые и административные параметры "в аренду" у центрального сервера, отвечающего за их распространение. Принцип аренды особенно удобен для персональных компьютеров, которые выключены, когда на них никто не работает, и провайдеров Internet, чьи клиенты подключаются по коммутируемым линиям.

К "арендуемым" параметрам относятся:

- IP-адреса и сетевые маски;
- адреса шлюзов (стандартные маршруты);
- адреса DNS-серверов;
- имена компьютеров, на которых выполняется система Syslog;
- адреса серверов WINS, NTP и прокси-серверов;
- адреса серверов TFTP (для получения файла начальной загрузки)

и десятки других (см. RFC2132). Экзотические параметры редко используются на практике. В большинстве случаев DHCP-сервер предоставляет лишь базовые сетевые параметры, такие как IP-адреса, сетевые маски, адреса стандартных шлюзов и адреса серверов DNS.

Периодически клиенты должны повторно обращаться к DHCP-серверу с целью продления срока аренды. Если этого не делать, аренда рано или поздно прекратится. DHCP-сервер будет тогда волен предоставить адрес (или иной арендованный параметр) другому клиенту. Срок аренды конфигурируется, но обычно он достаточно велик (до нескольких дней).

Протокол DHCP существенно облегчает жизнь системным администраторам. Когда сервер DHCP сконфигурирован и запущен, клиенты автоматически определяют параметры сетевой конфигурации на этапе начальной загрузки, и никакой путаницы не возникает.

Программное обеспечение DHCP

Программное обеспечение DHCP-серверов и клиентов в различных дистрибутивах Linux всегда в значительной мере различалось. В современных версиях оно более-менее стандартизовано и соответствует эталонной реализации, разработанной консорциумом Internet Systems Consortium (Консорциум разработчиков программного обеспечения для Internet) (ISC). Сервер ISC поддерживает также протокол BOOTP, который напоминает DHCP, но не так сложен. Во всех современных реализациях программное обеспечение DHCP-клиента устанавливается по умолчанию, но чтобы можно было использовать сервер ISC и агента ретрансляции, иногда может требоваться установка дополнительных пакетов.

Чтобы начать диалог с DHCP-сервером, DHCP-клиенты должны отправить пакет с общим широковещательным адресом (все единицы). Дело в том, что маска подсети на данном этапе еще не известна клиенту, поэтому использовать локальный широковещательный адрес не получится.

DHCP-сервер ISC понимает протокол динамического обновления базы данных DNS. Сервер не только предоставляет компьютеру IP-адрес и другие сетевые параметры, но также вносит в базу данных DNS запись о соответствии между именем хоста и его IP-

адресом. Дополнительная информация о динамических обновлениях приведена в разделе 15.14.

В следующих подразделах мы вкратце рассмотрим особенности протокола DHCP, инсталляцию сервера ISC, реализующего этот протокол, а также конфигурирование DHCP-клиентов.

Схема работы DHCP

DHCP — это расширение протокола BOOTP, который был придуман для того, чтобы бездисковые UNIX-станции могли загружаться по сети. Сервер BOOTP предоставляет клиентам IP-адреса, сетевые маски, адреса стандартных шлюзов, а также информацию, касающуюся начальной загрузки через TFTP (Trivial File Transfer Protocol — простейший протокол передачи файлов). Протокол DHCP не ограничивается этими параметрами, вводя понятие "аренды".

DHCP-клиент начинает диалог с DHCP-сервером, посылая широковещательное сообщение вида "помогите мне узнать, кто я". Если в локальной сети есть DHCP-сервер, он договаривается с клиентом об аренде IP-адреса и других сетевых параметров (сетевой маски, адреса сервера имен и стандартного шлюза). Если же такого сервера нет, то сообщение может быть передано в другие подсети через специальный прокси-сервер, называемый *агентом ретрансляции*.

По истечении половины срока аренды клиент должен ее продлить. Сервер обязан отслеживать адреса, предоставленные в аренду, и сохранять эту информацию при перезагрузке. Предполагается, что клиенты делают то же самое, хотя это не обязательно. Благодаря этому достигается максимальная стабилизация сетевой конфигурации.

Протокол DHCP обычно не используется для конфигурирования коммутируемых PPP-интерфейсов. Для этого существует специальный протокол PPPCP (PPP Control Protocol — протокол управления PPP-соединениями).

DHCP-сервер ISC

Для конфигурирования DHCP-сервера `dhcpd` нужно отредактировать файл `dhcpd.conf` в каталоге `server` и записать его под именем `/etc/dhcpd.conf`[20]. Необходимо также создать пустой файл базы данных по арендуемым параметрам, назвав его `/var/db/dhcp.leases` (подойдет команда `touch`). Убедитесь в том, что демон `dhcpd` имеет право записи в этот файл. Для заполнения файла `dhcpd.conf` потребуется следующая информация:

- адреса подсетей, для которых демон `dhcpd` должен управлять IP-адресами, и диапазоны выделяемых адресов;
- начальный и максимальный сроки аренды в секундах;
- конфигурационные параметры клиентов BOOTP, если таковые имеются (им назначаются статические IP-адреса, также должны быть указаны их аппаратные MAC-адреса);
- все остальные параметры, которые сервер должен передавать DHCP-клиентам: сетевая маска, стандартный маршрут, домен DNS, адреса серверов имен и т.д.

[20] Будьте осторожны! У файла `dhcpd.conf` очень "чувствительный" формат: достаточно удалить какую-нибудь точку с запятой, и будет выдано непонятное сообщение об ошибке.

На man-странице, посвященной демону dhcpd, дан обзор процесса конфигурации. Точный синтаксис конфигурационного файла описан на man-странице файла dhcpd.conf. Обе эти страницы расположены в подкаталоге server дистрибутива. В некоторых дистрибутивах в каталоге /etc есть образец файла dhcpd.conf. Его можно изменить в соответствии с локальными параметрами.

Демон dhcpd должен автоматически запускаться на этапе начальной загрузки системы. Удобно сделать запуск демона условным, осуществляемым при наличии файла /etc/dhcpd.conf.

Ниже показан пример файла dhcpd.conf, взятого из Linux-системы с двумя сетевыми интерфейсами: внутренним и внешним, подключенным к Internet. На компьютере выполняется система NAT для трансляции адресов внутренней сети, которой предоставляются десять IP-адресов. Файл содержит пустую запись для внешнего интерфейса (обязательна) и запись host для одного компьютера, которому нужен фиксированный адрес.

```
# dhcpd.conf
#
# глобальные параметры
option domain-name "synack.net";
option domain-name-servers gw.synack.net;
option subnet-mask 255.255.255.0;
default-lease-time 600;
max-lease-time 7200;
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.51 192.168.1.60;
    option broadcast-address 192.168.1.255;
    option routers gw.synack.net;
}

subnet 209.180.251.0 netmask 255.255.255.0 {
}
host gandalf {
    hardware ethernet 08:00:07:12:34:56;
    fixed-address gandalf.synack.net;
}
```

📖 Подробнее о DNS рассказывается в главе 15.

Адреса, назначаемые DHCP-сервером, могут конфликтовать с содержимым базы данных DNS. Очень часто каждому динамически выделяемому адресу присваивается общее имя (например, dhcp1.synack.net) и разрешается, чтобы имена отдельных компьютеров изменялись вместе с IP-адресами. Если используется одна из последних версий пакета BIND, в которой поддерживаются динамические обновления, то можно сконфигурировать демон dhcpd так, чтобы он обновлял базу данных DNS при выделении очередного адреса. Динамические обновления — сложное решение, но оно позволяет сохранять имена компьютеров неизменными.

Демон dhcpd фиксирует каждый акт аренды в файле dhcp.leases. Он также периодически создает резервную копию этого файла, переименовывая его в dhcp.leases~ и воссоздавая исходный файл dhcp.leases на основании информации из резидентной базы данных. Если в процессе такой операции произойдет сбой, останется только файл dhcp.leases~. В этом случае демон dhcpd откажется запускаться, и придется переименовывать файл вручную. *Нельзя* просто создать пустой файл dhcp.leases, иначе возникнет хаос, так как у клиентов окажутся дублирующиеся адреса.

DHCP-клиент, в общем-то, не требует конфигурирования. Файлы с информацией о статусе каждого соединения хранятся в каталоге /var/lib/dhcp или /var/lib/dhclient. Имя файла соответствует имени описываемого интерфейса. Например, файл dhclient-eth0.leases будет содержать все сетевые параметры, которые демон dhclient закрепил за интерфейсом eth0.

12.10. Динамическое переконфигурирование и настройка

В Linux настройка ядра и сетевых параметров осуществляется по-особому. Вместо того чтобы предоставить обычный конфигурационный файл, считываемый для определения соответствующих значений, разработчики Linux поместили "образ" каждой настраиваемой переменной в специальную файловую систему /proc. Важные сетевые параметры находятся в каталоге /proc/sys/net/ipv4:

```
$ cd /proc/sys/net/ipv4; ls -F
conf/                           ip_local_port_range    tcp_mem
icmp_echo_ignore_all            ip_nonlocal_bind       tcp_moderate_rcvbuf
icmp_echo_ignore_broadcasts     ip_no_pmtu_disc        tcp_no_metrics_save
icmp_errors_use_inbound_ifaddr  neigh/                 tcp_orphan_retries
icmp_ignore_bogus_error_responsesroute/                tcp_reordering
icmp_ratelimit                  tcp_abc                tcp_retrans_collapse
icmp_ratemask                   tcp_abort_on_overflow  tcp_retries1
igmp_max_memberships            tcp_adv_win_scale      tcp_retries2
igmp_max_msf                    tcp_app_win            tcp_rfc1337
inet_peer_gc_maxtime            tcp_congestion_control tcp_rmem
inet_peer_gc_mintime            tcp_dma_copybreak      tcp_sack
inet_peer_maxttl                tcp_dsack              tcp_stdurg
inet_peer_minttl                tcp_ecn                tcp_synack_retries
inet_peer_threshold             tcp_fack               tcp_syncookies
ip_autoconfig                   tcp_fin_timeout        tcp_syn_retries
ip_default_ttl                  tcp_frto               tcp_timestamps
ip_dynaddr                      tcp_keepalive_intvl    tcp_tso_win_divisor
ip_forward                      tcp_keepalive_probes   tcp_tw_recycle
ipfrag_high_thresh              tcp_keepalive_time     tcp_tw_reuse
ipfrag_low_thresh               tcp_low_latency        tcp_window_scaling
ipfrag_max_dist                 tcp_max_orphans        tcp_wmem
ipfrag_secret_interval          tcp_max_syn_backlog
ipfrag_time                     tcp_max_tw_buckets
```

Многие переменные, в именах которых присутствуют компоненты "rate" и "max", используются для пресечения атак вида "отказ в обслуживании". В подкаталоге conf содержатся переменные, устанавливаемые отдельно для каждого интерфейса. Там есть подкаталоги all и default, а также подкаталоги интерфейсов (включая интерфейс обратной связи). В каждом подкаталоге представлен одинаковый набор файлов:

```
$ cd conf/default; ls -F
accept_redirects    bootp_relay           log_martiansrp_filter
accept_source_route disable_policy        mc_forwardingsecure_redirects
arp_announce        disable_xfrm          medium_idsend_redirects
arp_filter          force_igmp_version    promote_secondariesshared_media
arp_ignore          forwarding            proxy_arptag
```

Изменения элементов подкаталога `all` отражаются на всех интерфейсах. Но если та же переменная меняется, скажем, в подкаталоге `eth0`, то это затронет только соответствующий интерфейс. В подкаталоге `default` хранятся стандартные значения переменных на момент инсталляции системы.

Каталог `neigh` содержит подкаталоги для каждого интерфейса. Файлы этих подкаталогов задают параметры управления таблицей ARP и обнаружения соседей IPv6 для данного интерфейса. Ниже приведен список содержимого одного из таких каталогов. Файлы с именами на `gc_` (от "garbage collection" — сборка мусора) определяют параметры устаревания и удаления записей таблицы ARP.

```
$ cd neigh/default; ls -F
anycast_delay           gc_interval       locktimeretrans_time_ms
app_solicit             gc_stale_time     mcast_solicitucast_solicit
base_reachable_time     gc_thresh1        proxy_delayunres_qlen
base_reachable_time_ms  gc_thresh2        proxy_qlen
delay_first_probe_time  gc_thresh3        retrans_time
```

Чтобы узнать значение переменной, воспользуйтесь обычной командой `cat`. Изменить значение переменной можно с помощью команды `echo`, перенаправив результаты ее работы в соответствующий файл. Например, команда

```
$ cat icmp_echo_ignore_broadcasts
0
```

показывает, что широковещательные ICMP-пакеты допустимы. Чтобы защитить систему от атак типа "smurf", введите

```
$ sudo sh -c "echo 1 > icmp_echo_ignore_broadcasts"[21]
```

в каталоге `/proc/sys/net`. Будьте осторожны при изменении настроек сети, через которую сами зарегистрировались в системе. В случае ошибки иногда оказывается, что перезагрузка системы возможна только с консоли.

Чтобы изменить любой из параметров на постоянной основе (точнее, чтобы переустанавливать его всякий раз при перезагрузке системы), добавьте соответствующие переменные в файл `/etc/sysctl.conf`, считываемый командой `sysctl` в процессе начальной загрузки. Формат записей файла `sysctl.conf` таков: *`переменная`*`=`*`значение`* (в отличие от команды `echo` *`значение`* `>` *`переменная`*, запускаемой из интерпретатора команд для изменения переменной вручную). Имена переменных даются относительно каталога `/proc/sys`. Вместо символов косой черты разрешается использовать точки. Например, любая из записей

```
net.ipv4.ip_forward=0
net/ipv4/ip_forward=0
```

в файле `/etc/sysctl.conf` приведет к отключению механизма перенаправления IP-пакетов (на данном хосте).

[21] Если задать команду `sudo echo 1 > icmp_echo_ignore_broadcasts`, будет получено сообщение "permission denied" (доступ запрещен), поскольку интерпретатор команд попытается открыть выходной файл до запуска утилиты `sudo`. В данном случае необходимо, чтобы действие утилиты `sudo` распространялось не только на команду `echo`, но и на операцию переадресации. Следовательно, нужно создать вторичный интерпретатор с правами пользователя `root`, в котором и будет выполнена вся команда.

В документе /usr/src/linux/Documentation/proc.txt, написанном разработчиками SUSE, приведены примеры настройки ядра средствами файловой системы /proc[22]. В документе объясняется, что означает та или иная переменная, и в некоторых случаях указывается, какие значения лучше всего задавать. Кроме того, файл немного устарел, поскольку авторы документации не всегда успевают угнаться за разработчиками ядер.

12.11. Безопасность

Теме безопасности будет посвящена отдельная глава (глава 20), но ряд вопросов, касающихся IP-сетей, заслуживает отдельного упоминания. В этом разделе мы рассмотрим сетевые механизмы, которые традиционно вызывают проблемы безопасности, и опишем пути решения этих проблем.

Перенаправление IP-пакетов

Если в Linux-системе разрешено перенаправление IP-пакетов, то компьютер может выступать в качестве маршрутизатора. Активизировать эту функцию рекомендуется, если в системе имеется несколько сетевых интерфейсов и она действительно играет роль уполномоченного сетевого маршрутизатора. Посредством механизма перенаправления можно сделать так, что внешние пакеты будут казаться поступившими от локального компьютера. Эта уловка позволяет пакетам хакеров обходить сетевые сканеры и фильтры.

Переадресующие ICMP-пакеты

С помощью переадресующих ICMP-пакетов можно злонамеренно менять направление трафика и редактировать таблицы маршрутизации. Большинство операционных систем по умолчанию принимает эти пакеты и следует содержащимся в них инструкциям. Но, согласитесь, вряд ли можно назвать приемлемой ситуацию, когда на несколько часов весь трафик организации перенаправляется конкуренту, особенно если в это время выполняется резервное копирование. Мы рекомендуем так конфигурировать маршрутизаторы (или системы, играющие роль маршрутизаторов), чтобы переадресующие ICMP-пакеты игнорировались и, возможно, фиксировались в журнальном файле.

Маршрутизация "от источника"

Механизм маршрутизации "от источника" в протоколе IP позволяет отправителю явно указать последовательность шлюзов, через которые должен пройти пакет на пути к получателю. При этом отключается алгоритм поиска следующего перехода, выполняемый на каждом шлюзе для определения того, куда нужно послать пакет.

Маршрутизация "от источника" была частью исходной спецификации протокола IP и служила для целей тестирования. Но она создает проблему с точки зрения безопасности, ведь пакеты часто фильтруются в зависимости от того, откуда они прибыли. Злоумышленник может так подобрать маршрут, что пакет будет казаться прибывшим из внутренней сети, а не из Internet, поэтому брандмауэр его пропустит. Мы рекомендуем не принимать и не перенаправлять доставленные подобным образом пакеты.

[22] Чтобы копия этого файла стала доступной, необходимо инсталлировать пакет исходных кодов ядра.

Широковещательные ICMP-пакеты и другие виды направленных широковещательных сообщений

Пакеты команды `ping`, несущие в себе широковещательный адрес сети (а не конкретного хоста), обычно доставляются всем хостам сети. Такие пакеты применяются в атаках типа "отказ в обслуживании", например в так называемых атаках "smurf" (по названию программы, в которой они впервые были применены). Большинство хостов располагает средствами запрета широковещательных ICMP-запросов, т.е. их можно сконфигурировать на запрет приема и перенаправления таких пакетов. Маршрутизатор также способен фильтровать широковещательные пакеты, поступающие из Internet, чтобы они не попадали в локальную сеть. Лучше всего устанавливать средства защиты как на самом компьютере, так и на брандмауэре, если это возможно.

Широковещательные ICMP-пакеты одновременно являются "направленными" в том смысле, что они посылаются по широковещательному адресу конкретной удаленной сети. Стандартные подходы к обработке таких пакетов постепенно меняются. Например, в операционной системе Cisco IOS 11.x и более ранних версий по умолчанию осуществлялась переадресация направленных широковещательных пакетов, но в версиях 12.0 и выше этого уже нет. Обычно можно настроить стек TCP/IP так, чтобы широковещательные пакеты, приходящие из других сетей, игнорировались, но, поскольку это нужно сделать для каждого сетевого интерфейса, подобная задача является весьма нетривиальной в крупной организации.

Подмена IP-адресов

Исходный адрес IP-пакета обычно заполняется реализацией стека протоколов TCP/IP ядра и представляет собой адрес хоста, с которого был отправлен пакет. Но если программа, создающая пакет, работает с низкоуровневым IP-сокетом, она может подставить любой исходный адрес, какой пожелает. Это называется *подменой IP-адресов* и обычно ассоциируется с попытками взлома сети. В этой схеме жертвой часто становится компьютер, идентифицируемый по поддельному IP-адресу (если он действительно существует). Сообщения об ошибках и ответные пакеты могут переполнить и вывести из строя сеть жертвы.

Поддельные IP-адреса следует запрещать на пограничном маршрутизаторе, блокируя отправляемые пакеты, исходные адреса которых не находятся в подконтрольном диапазоне. Это особенно важно в университетских сетях, где студенты любят экспериментировать и часто подобным образом срывают свою злость.

В то же время, если в локальной сети используются адреса из частного диапазона, то фильтрации могут подвергаться пакеты с частными адресами, пытающиеся "проскочить" в Internet. Ответ на такие пакеты никогда не будет получен из-за отсутствия соответствующих маршрутов в магистральной сети. Их появление свидетельствует о том, что в системе имеется внутренняя ошибка конфигурации.

Описываемые ниже встроенные брандмауэры Linux позволяют реализовать такой вид фильтрации на каждом локальном хосте. Тем не менее, в большинстве организаций это делается лишь на пограничных маршрутизаторах. Мы рекомендуем именно такой подход. Встроенные брандмауэры могут понадобиться лишь в особых случаях.

Нужно также защищаться от хакеров, подделывающих исходные адреса внешних пакетов, вследствие чего брандмауэр начинает считать, будто они поступили из локальной сети. Выявление таких пакетов контролируется параметром ядра `rp_filter` (устанав-

ливается в каталоге `/proc/sys/net/ipv4/conf/`*интерфейс*), где префикс `rp` означает "reverse path" (обратный путь). Если этот параметр равен 1, ядро будет удалять все пакеты, удовлетворяющие следующему условию: интерфейс, через который пришел пакет, не совпадает с тем интерфейсом, через который пакет уйдет, если его исходный адрес будет равен целевому адресу. Этот режим включен по умолчанию.

Когда в организации есть несколько выходов в Internet, имеет смысл разделить внешние маршруты на исходящие и входящие. В этом случае параметр `rp_filter` нужно сделать равным нулю, чтобы протокол маршрутизации работал правильно. Если же выход в Internet только один, безопаснее установить параметр `rp_filter` равным единице.

Встроенные брандмауэры Linux

В Linux имеются встроенные средства фильтрации пакетов (брандмауэры). Мы опишем эти средства чуть ниже (раздел 12.12), а также в главе 20 (разделы 20.12), хотя и не рекомендуем использовать рабочую станцию в качестве брандмауэра. Безопасность Linux-систем (особенно в том виде, в каком они выпускаются нашими любимыми поставщиками) и так невысока, а безопасность Windows-систем — еще хуже, поэтому лучше приобрести специализированный брандмауэр. Даже самые сложные программные решения, например система Firewall-1 компании Check Point (работает в Solaris), уступают в надежности специализированным аналогам, в частности устройствам PIX компании Cisco, хотя имеют почти такую же цену!

Подробнее вопросы применения брандмауэров рассмотрены в разделе 20.12.

Виртуальные частные сети

Многим организациям, имеющим офисы в различных частях света, хотелось бы, чтобы все эти офисы были соединены одной большой частной сетью. К сожалению, стоимость аренды трансконтинентальных и даже транснациональных линий связи делает это нереальным. Таким организациям приходится использовать Internet в качестве "частного" канала, организуя серию защищенных, зашифрованных "туннелей" между офисами. Сетевые конгломераты подобного рода называются *виртуальными частными сетями*.

О протоколе IPSEC рассказывается в разделе 20.14.

В ряде частных сетей используется протокол IPSEC, который в 1998 г. был стандартизирован организацией IETF. В других сетях применяются собственные патентованные решения, как правило, несовместимые друг с другом. Мы рекомендуем обратить внимание на такие продукты, как маршрутизатор 3660 компании Cisco и брандмауэр Firebox компании Watchguard; оба они поддерживают туннелирование и шифрование. Устройство Firebox использует протокол PPP для управления последовательным портом. Системный администратор может установить удаленное соединение с этим устройством, чтобы сконфигурировать его или получить доступ в виртуальную частную сеть для тестирования.

Что касается недорогих решений, то в разделе 12.13 приведен пример, в котором посредством команды `ssh` организуется PPP-канал, реализующий виртуальную частную сеть.

Переменные ядра, связанные с безопасностью

В табл. 12.15 описано стандартное поведение Linux-систем в отношении различных сетевых методик обеспечения безопасности. Все они были кратко рассмотрены выше. Мы рекомендуем установить соответствующие параметры так, чтобы система не отвечала на широковещательные ICMP-запросы, не подчинялась директивам переадресации и не принимала пакеты, маршрутизируемые "от источника".

Таблица 12.15. Реализация методик, связанных с сетевой безопасностью, в Linux

Методика	на простом хосте	на шлюзе	Управляющий файл (в каталоге /proc/sys/net)
Перенаправление IP-пакетов	Отключено	Включено	ipv4/ip_forward для всей системы ipv4/conf/*интерфейс*/forwarding для каждого интерфейса [a]
Переадресующие ICMP-пакеты	Разрешена	Не разрешена	ipv4/conf/*интерфейс*/accept_redirects
Маршрутизация "от источника"	Не разрешена	Разрешена	ipv4/conf/*интерфейс*/accept_source_route
Широковещательные ICMP-пакеты	Принимаются	Принимаются	ipv4/icmp_echo_ignore_broadcasts

[a] В качестве параметра *интерфейс* может быть задано имя конкретного интерфейса или ключевое слово all.

12.12. Реализация системы NAT в Linux

В Linux традиционно реализуется лишь ограниченная разновидность системы NAT, называемая PAT (Port Address Translation — трансляция адресов портов). В ней не используется диапазон частных адресов, как в истинной системе NAT, а все соединения коммутируются по одному адресу. Чтобы все выглядело еще запутаннее, в документации к Linux часто говорится не о NAT или PAT, а об "IP-маскировании" (IP masquerading). С практической точки зрения это не имеет особого значения, поэтому для простоты мы будем продолжать употреблять термин "NAT".

iptables не только реализует систему NAT, но и обладает средствами фильтрации пакетов. В первых версиях Linux выполнять фильтрацию было сложновато, но в команде iptables проведено более четкое разграничение между системой NAT и фильтрацией пакетов.

О принципах фильтрации будет подробнее рассказываться в разделах 20.12 и 21.13. Чтобы систему NAT можно было использовать для обеспечения доступа локальных хостов к Internet, вместе с ней придется использовать полный набор фильтров брандмауэра. То, что система NAT "не является действительным механизмом IP-маршрутизации" не делает шлюзы NAT Linux более безопасными, чем маршрутизаторы Linux. Для краткости мы рассмотрим конфигурирование только самой системы NAT, но следует помнить, что это — лишь незначительная часть полной конфигурации.

Чтобы система NAT могла успешно работать, необходимо включить перенаправление IP-пакетов в ядре, установив значение переменной ядра /proc/sys/net/ipv4/ip_forward равной 1. Кроме того, необходимо вставить соответствующие модули ядра:

```
$ sudo /sbin/modprobe iptable_nat
$ sudo /sbin/modprobe ip_conntrack
$ sudo /sbin/modprobe ip_conntrack_ftp
```

Команда `iptables`, включающая маршрутизацию пакетов посредством системы NAT, имеет следующую форму:

```
$ sudo iptables -t nat -A POSTROUTING -o eth1 -j SNAT --to 63.173.189.1
```

В этом примере `eth0` — интерфейс, подключенный к Internet, а его адрес определен в аргументе `--to`. Интерфейс `eth1` подключен к внутренней сети.

Хосты Internet воспринимают все пакеты, поступающие от хостов внутренней сети, как поступающие из IP-адреса интерфейса `eth0`. Хост, на котором выполняется система NAT, принимает поступающие пакеты, выясняет их истинный адрес назначения, перезаписывает их с соответствующим IP-адресом внутренней сети и пересылает по месту назначения.

12.13. PPP: протокол двухточечного соединения

PPP (Point-to-Point Protocol — протокол двухточечного соединения) применяется как в самых медленных, так и в самых быстрых Internet-соединениях. В синхронной форме это протокол инкапсуляции пакетов, используемый в высокоскоростных соединениях, на обоих концах которых установлены мощные маршрутизаторы. В асинхронной форме это протокол, определяющий правила кодирования IP-пакетов с целью передачи по медленному (и часто ненадежному) последовательному каналу. Каналы такого типа просто ретранслируют потоки битов и не "знают" о том, где начинается и заканчивается пакет. За кодирование и декодирование пакетов, передаваемых по последовательному каналу, отвечает драйвер PPP.

Он добавляет к пакету заголовок канального уровня и маркеры-разграничители. Иногда протокол PPP используется в новых сетевых устройствах домашнего применения, таких как DSL и кабельные модемы, но, как правило, это обстоятельство вне сферы ответственности системных администраторов. Инкапсуляция соответствующих функций выполняется интерфейсным устройством, а трафик переадресуется в сеть Ethernet. Пользователь же видит только Ethernet-подключение.

PPP — это универсальный протокол инкапсуляции, разработанный комитетом по стандартизации. При его создании были использованы идеи, заложенные в протоколы SLIP (Serial Line IP — IP-протокол последовательной линии), разработанный Риком Адамсом (Rick Adams), и CSLIP (Compressed SLIP — IP-протокол последовательной линии со сжатием), разработанный Ван Якобсоном (Van Jacobson). Протокол PPP отличается от них тем, что позволяет передавать пакеты множества протоколов по одному каналу. Спецификация протокола приведена в документе RFC1331.

Производительность протокола PPP

Протокол PPP обеспечивает все функциональные возможности сетей Ethernet, но на гораздо меньшей скорости. Обычные офисные локальные сети работают со скоростью 100 или 1000 Мбит/с, т.е. 100 000–1 000 000 Кбит/с. А коммутируемое соединение функционирует со скоростью 28–56 Кбит/с. Передача 1-Мбайтного файла по коммутируемой линии, работающей под управлением протокола PPP, занимает около 5 минут. Это еще можно стерпеть при отправке электронной почты или при работе с Web-сайтами в режиме без отображения рисунков, но просмотр Web-сайтов в обычном режиме будет стоить очень много нервов. Для увеличения скорости можно сделать параметр MTU (максимальный размер передаваемого блока) PPP-соединения очень низким. По умолчанию

он равен 512 байт; задайте 128, если выполняется много интерактивной работы. Если PPP-соединение функционирует поверх сети Ethernet, то узнать размер передаваемых пакетов позволяет утилита `tcpdump`. В Ethernet значение MTU равно 1500, но наличие PPP-инкапсуляции приводит к тому, что более эффективными становятся чуть меньшие значения. Например, утилита `pppoe` предполагает, что хосты, находящиеся за пределами PPP-соединения, работают с пакетами размером 1412 байт, а в самом PPP-канале размер пакетов составляет 1492 байт. Конечно, нежелательно, чтобы из-за подобной разницы пакеты подвергались фрагментации, поэтому стандартное значение MTU лучше изменить.

📖 Подробнее об NFS рассказывается в главе 16.

Работать с NFS через PPP-соединение очень неудобно из-за низкой скорости. Имеет смысл делать это только в том случае, когда в NFS используется протокол TCP, а не UDP.

В X Window используется именно протокол TCP, поэтому через PPP-соединение можно запускать X-приложения. Программы наподобие `xterm` работают прекрасно, но следует избегать приложений с экзотическими шрифтами и растровой графикой.

Подключение к сети посредством протокола PPP

Для того чтобы компьютер можно было подключить к сети средствами PPP, предварительно нужно обеспечить выполнение трех условий.

- Ядро компьютера должно быть способно посылать IP-пакеты через последовательный канал в соответствии с требованиями протокола PPP.
- В системе должна существовать программа пользовательского уровня, позволяющая устанавливать и обслуживать PPP-соединения.
- Система на другом конце последовательного канала должна понимать используемый протокол.

Как заставить систему работать по протоколу PPP

📖 Подробнее о команде `ifconfig` рассказывается в разделе 12.7.

Для установки PPP-соединения необходимо, чтобы система могла посылать и принимать PPP-пакеты. В Linux протокол PPP реализован в виде загружаемого модуля ядра, который помещает сетевые пакеты в выходную очередь устройства последовательного доступа и извлекает поступающие пакеты из входной очереди. Обычно этот модуль функционирует подобно сетевому интерфейсу, поэтому им можно манипулировать с помощью стандартных средств, например команды `ifconfig`.

Управление PPP-соединениями

Точная последовательность событий, связанных с установлением PPP-соединения, зависит от операционной системы и типа сервера, с которым осуществляется связь. Соединения можно устанавливать либо вручную, либо динамически.

При ручном подключении пользователь запускает команду, которая звонит по модему, входит в удаленную систему и активизирует в ней подсистему PPP. Если эта процедура заканчивается успешно, последовательный порт конфигурируется как сетевой интерфейс. В данном варианте соединение обычно остается активным длительное время, что лучше всего подходит для выделенных телефонных линий.

В динамической конфигурации демон наблюдает за последовательными "сетевыми" интерфейсами, отслеживая запросы на передачу данных через них. Когда кто-то пытается послать пакет, демон автоматически звонит по модему для установления соединения, передает пакет и, если канал возвращается в неактивное состояние, по истечении определенного времени разрывает соединение. Динамические коммутируемые соединения применяются, когда по телефонной линии передаются как голосовые, так и двоичные данные либо когда взимается плата за междугородную связь или за время пользования каналом.

Программы, реализующие обе эти схемы, входят в состав большинства реализаций протокола PPP.

Присвоение адреса

📖 Подробнее о назначении IP-адресов рассказывается в разделе 12.7.

Каждому PPP-интерфейсу должен быть назначен IP-адрес, аналогично тому, как это имеет место при включении нового компьютера в сеть Ethernet. Существует несколько способов присвоения адресов PPP-каналам (иногда для установления связи вообще не требуются IP-адреса). Ниже рассмотрен самый простой из них.

PPP-канал следует рассматривать как отдельную сеть, состоящую из двух компьютеров (ее часто называют "двухточечной" сетью). Сетевой адрес присваивается каналу так же, как и новому сегменту Ethernet, в соответствии с правилами, принятыми в организации. В этой сети можно выбрать два любых адреса и присвоить по одному из них каждому концу канала. Можно также выполнить и другие локальные настройки, например, в соответствии со стандартами подсети. Каждый из компьютеров PPP-соединения становится для внешнего мира "шлюзом" к рассматриваемой двухточечной сети. (На практике пользователь не контролирует ни один из концов канала: используемые адреса предоставляются ему провайдером.)

Протокол DHCP может также присвоить IP-адрес концу PPP-соединения. Провайдеры Internet обычно предоставляют домашним клиентам услуги DHCP, а организациям выделяют статические адреса.

Маршрутизация

📖 Подробнее о маршрутизации рассказывается в главе 13.

Поскольку протокол PPP требует, чтобы удаленный сервер выступал в качестве маршрутизатора, вопросы IP-маршрутизации должны учитываться так же, как и на реальном шлюзе (например, на хосте, соединяющем две сети Ethernet). Задача маршрутизатора заключается в направлении пакетов по шлюзам в заданные пункты назначения. Настраивать схему маршрутизации можно разными способами.

На рядовом PPP-клиенте должен быть задан маршрут по умолчанию, обеспечивающий передачу пакетов PPP-серверу. Сервер должен быть известен остальным хостам сети как шлюз к клиенту.

Большинство PPP-систем решает эти задачи автоматически.

Безопасность

📖 Теме компьютерной безопасности посвящена глава 20.

Проблемы безопасности возникают всякий раз, когда в сети появляется новое устройство. Поскольку компьютер, подключенный средствами PPP, является полноправ-

ным членом сети, с ним нужно обращаться должным образом: следить, чтобы не было учетных записей без паролей или с ненадежными паролями, чтобы были инсталлированы все средства защиты и т.д. Ряд вопросов сетевой безопасности рассматривался в разделе 12.11. В Linux поддерживаются два протокола аутентификации PPP-соединений: PAP (Password Authentication Protocol — протокол парольной аутентификации) и CHAP (Challenge Handshake Authentication Protocol — протокол аутентификации с предварительным квитированием).

Диалоговые сценарии

В Linux при организации последовательного PPP-соединения выполняется диалоговый сценарий, предназначенный для установки модемного соединения, регистрации на удаленном компьютере и запуска PPP-сервера. Этот сценарий состоит из набора посылаемых строк и строк, которые предполагается получить в ответ. Имеется условная конструкция, позволяющая задавать правила вида "ожидать строку 'Login', но если она не будет получена — послать символ возврата каретки и подождать еще".

Диалоговые сценарии впервые появились в системе UUCP, и было это давным-давно. В 1980-х гг. компьютеры могли связаться друг с другом посреди ночи, зарегистрироваться с помощью диалоговых сценариев и произвести обмен файлами. Несмотря на бытующее мнение, система UUCP не совсем исчезла: в SUSE группе `uucp` принадлежат файлы устройств последовательного доступа, и нужно быть членом этой группы, чтобы получить возможность устанавливать исходящие модемные PPP-соединения.

В большинстве реализаций PPP существуют готовые диалоговые сценарии, которые можно адаптировать к конкретной конфигурации. В сценарии нужно задать такие параметры, как номер телефона, по которому делается звонок, и команда, запускаемая после успешного входа в систему. В большинстве сценариев содержится незашифрованный пароль, поэтому следует соответствующим образом установить права доступа к ним.

Конфигурирование протокола PPP в Linux

Модемы (а также принтеры) всегда были головной болью системных администраторов. И это не удивительно, ведь программа настройки PPP-соединения для модема имеет более 125 возможных опций. Как тут не схватиться за голову!

В состав всех дистрибутивов за исключением Debian входит PPP-пакет Пола Макерраса (Paul Mackerras), в котором используется демон `pppd`, а большинство конфигурационных файлов хранится в каталоге `/etc/ppp`. Чтобы узнать, какая версия пакета инсталлирована в используемом дистрибутиве, выполните команду `pppd --version`. Для установки этого пакте в системе Debian необходимо использовать команду `apt-get install ppp`.

В рассматриваемые нами системы включен также PPP-пакет компании Roaring Penguin Software, предназначенный для работы поверх Ethernet (например, в DSL-соединении с локальным провайдером). Во всех демонстрационных дистрибутивах есть и средства организации PPP-соединений в сетях ISDN. Конфигурационные файлы этих программных средств тоже хранятся в каталоге `/etc/ppp`. Их можно узнать по наличию компонента `oe` (от "over Ethernet" — поверх Ethernet) или `i` (от ISDN). В табл. 12.16 приведены команды и файлы конфигурирования сети, используемые в рассматриваемых системах.

Таблица 12.16. Команды и конфигурационные файлы, относящиеся к протоколу PPP

Система	Команды и конфигурационные файлы	Описание
Все	/usr/sbin/pppd	Демон PPP
	/usr/sbin/chat	Обмен данными с модемом
	/usr/sbin/pppstats	Отображает статистику PPP-соединения
	/usr/sbin/pppdump	Позволяет просматривать PPP-пакеты в текстовом виде
	/etc/ppp/options	Конфигурационный файл демона pppd
Debian, Ubuntu	/usr/bin/pon	Устанавливает PPP-соединение
	/usr/bin/poff	Разрывает PPP-соединение
	/usr/bin/plog	Отображает заданный фрагмент файла ppp.log
	/usr/sbin/pppconfig	Конфигурирует демон pppd
	/etc/ppp/peers/provider	Содержит параметры подключения к провайдеру для программы pon
	/etc/chatscripts/provider	Диалоговый сценарий подключения к провайдеру для программы pon
Red Hat (DSL)	/usr/sbin/pppoe	Клиент PPP-соединений, устанавливаемых поверх Ethernet
	/usr/sbin/pppoe-server	Сервер PPP-соединений, устанавливаемых поверх Ethernet
	/usr/sbin/pppoe-sniff	Анализатор пакетов для отладки PPP-соединений
	/usr/sbin/adsl-connect	Сценарий для управления каналом
	/usr/sbin/adsl-setup	Сценарий настройки программы pppoe
	/usr/sbin/adsl-start	Сценарий активизации соединения для программы pppoe
	/usr/sbin/adsl-stop	Сценарий разрыва соединения для программы pppoe
	/usr/sbin/adsl-status	Отображает статус соединения
	/etc/ppp/pppoe.conf	Конфигурационный файл, используемый сценариями adsl-*
	/etc/ppp/pppoe-server-options	Файл, содержащий дополнительные параметры сервера
SUSE (DSL)	/usr/sbin/pppoed	Клиент PPP-соединений, устанавливаемых поверх Ethernet
	/etc/pppoed.conf	Конфигурационный файл программы pppoed
Все (ISDN)	/usr/sbin/ipppd	Демон PPP-соединений в сетях ISDN
	/usr/sbin/ipppstats	Отображает статистику PPP-соединения
	/etc/ppp/ioptions	Содержит параметры демона ipppd

📖 Более подробно имена файлов последовательных портов рассматриваются в разделе 27.6.

В приводимых ниже примерах конфигурационных файлов файл последовательного порта, к которому подключен модем, называется /dev/modem. В некоторых дистрибутивах действительно есть такой файл, являющийся ссылкой на один из соответствующих системных файлов (обычно /dev/ttyS0 или /dev/ttyS1). Тем не менее, этот подход в настоящее время не практикуется, поэтому имя файла необходимо заменить согласно установкам системы.

Помимо PPP-пакетов в каждом дистрибутиве присутствует программа `wvdial`, позволяющая набирать телефонный номер для установки соединения.

Ранее мы упомянули о файлах модемных портов и программах набора номера; а теперь давайте поговорим о том, как настроить демон `pppd` на работу с ними. Глобальные параметры демона задаются в файле `/etc/ppp/options`, а параметры конкретных соединений могут храниться в каталогах `/etc/ppp/peers` и `/etc/chatscripts` (в Debian и Ubuntu). В Red Hat, Fedora и SUSE диалоговые сценарии обычно помещаются в каталог `/etc/ppp` под именами `chat.`*`удаленный_хост`*. С другой стороны, в Red Hat файл `/etc/sysconfig/network-scripts/ifcfg-`*`терминал`* может содержать параметры соединений для конкретного PPP-интерфейса.

По умолчанию демон `pppd` сначала проверяет файл `options`, потом личный пользовательский файл `~/.ppprc`, затем файл конкретного соединения `options.`*`терминал`* (если он существует) и, наконец, аргументы своей командной строки.

Удобный прием, предложенный признанным авторитетом по Linux Джонатаном Корбетом (Jonathan Corbet), заключается в определении нескольких PPP-интерфейсов: одного для домашнего компьютера, другого для подключения к сети в отелях во время путешествий и т.д. Такой подход облегчает сетевые настройки в соответствии с конкретной ситуацией.

Утилита `wvdial` по умолчанию ведет себя "умнее", чем `chat`, если оставить ее параметры без изменений. Конфигурационная информация (характеристики модема, регистрационное имя, пароль, номер телефона и т.д.) берется из файла `/etc/wvdial.conf`. В одном конфигурационном файле можно задать параметры для нескольких адресатов. Определить характеристики модема позволяет утилита `wvdialconf`. С ее помощью можно создать начальную версию файла `wvdial.conf`.

Приводимые далее конфигурационные файлы взяты из нескольких источников. Первый файл, `/etc/ppp/options`, задает глобальные параметры демона `pppd`. Ниже показано содержимое этого файла в каждом из наших дистрибутивов.

В Red Hat и Fedora:

```
lock
```

В SUSE:

```
noipdefault
noauth
crtscts
lock
modem
asyncmap 0
nodetach
lcp-echo-interval 30
lcp-echo-failure 4
lcp-max-configure 60
lcp-restart 2
idle 600
noipx
file /etc/ppp/filters
```

В Debian и Ubuntu:

```
asyncmap 0
```

```
auth
crtscts
lock
hide-password
modem
proxyarp
lcp-echo-interval 30
lcp-echo-failure 4
noipx
```

Мы предпочитаем использовать следующий файл глобальных параметров:

```
# Глобальные параметры PPP-соединений.
lock                # Всегда блокировать используемое устройство
asyncmap 0x00000000 # По умолчанию не использовать Escape-коды
crtscts             # Использовать аппаратный контроль передачи
defaultroute        # Добавить стандартный маршрут через PPP-интерфейс
mru 552             # MRU/MTU = 512 (данные) + 40 (заголовок)
mtu 552
```

Ниже показан образец файла `/etc/sysconfig/network-scripts/ifcfg-ppp0` в Red Hat. Он был создан утилитой `linuxconf`:

```
PERSIST=yes
DEFROUTE=yes
ONBOOT=no
INITSTRING=ATZ
MODEMPORT=/dev/modem
LINESPEED=115200
ESCAPECHARS=no
DEFABORT=yes
HARDFLOWCTL=yes
DEVICE=ppp0
PPPOPTIONS=
DEBUG=yes
PAPNAME=remote
REMIP=
IPADDR=
BOOTPROTO=none
MTU=
MRU=
DISCONNECTTIMEOUT=
RETRYTIMEOUT=
USERCTL=no
```

А вот пример диалогового сценария (`chat.ppp0`), соответствующего показанному выше файлу `ifcfg-ppp0`:

```
'ABORT' 'BUSY'
'ABORT' 'ERROR'
'ABORT' 'NO CARRIER'
'ABORT' 'NO DIALTONE'
'ABORT' 'Invalid Login'
'ABORT' 'Login incorrect'
'' 'ATZ'
'OK' 'ATDT телефонный_номер'
'CONNECT' ''
```

```
'TIMEOUT' '120'
'ogin:' 'регистрационное_имя'
'ord:' 'пароль'
'TIMEOUT' '5'
'~--' ''
```

Некоторые строки этого сценария содержат пустой параметр, заключенный в пару одинарных кавычек, которые, в зависимости от используемого шрифта могут выглядеть как символы двойных кавычек.

Обычно диалоговый сценарий можно адаптировать к конкретным условиям, не вникая в то, как он работает. В данном случае первые несколько строк определяют, в каких случаях сценарий должен прекращать работу. В следующих строках происходит инициализация модема и набор телефонного номера, после чего вводятся соответствующие имя пользователя и пароль.

Иногда приходится менять продолжительность тайм-аута, чтобы приспособиться к особенностям работы местной АТС. Большинство модемов понимает запятую в телефонном номере как признак паузы при наборе. Это важно, если нужно набрать определенную цифру и подождать секунду, пока не появится сигнал.

В нашей системе регистрационные имена PPP соответствуют именам пользователей, к которым добавлен префикс "P". Это позволяет определять, кому принадлежит конкретное PPP-соединение.

Связь между файлами `ifcfg-ppp0` и `chat.ppp0` устанавливается командой `ifup`, которая автоматически вызывается на этапе начальной загрузки, если существует файл `ifcfg-*`. Можно также вызвать демон `pppd` явно, передав ему в качестве аргумента конфигурационный файл, при условии, что в этом файле есть строка `connect`, где указан соответствующий диалоговый сценарий.

Следующий пример взят из Debian. В этой системе параметры соединений хранятся в каталоге `peers`, диалоговый сценарий — в каталоге `/etc/chatscripts`, а вместо обычных паролей используется механизм аутентификации PAP. Сначала приведем содержимое конфигурационного файла `/etc/ppp/peers/my-isp`:

```
/dev/модем              ### подставьте имя файла модемного порта
debug
crtscts
name имя_пользователя   ### имя пользователя на хосте провайдера
remotename провайдер
noauth
noipdefault
defaultroute
connect '/usr/sbin/chat -v -f /etc/chatscripts/my-isp'
```

Файл `/etc/chatscripts/my-isp` содержит такие записи:

```
'ABORT' 'BUSY'
'ABORT' 'ERROR'
'ABORT' 'NO CARRIER'
'ABORT' 'NO DIALTONE'
'' 'ATZ'
'OK' 'ATDT телефонный_номер'
'CONNECT' ''
'TIMEOUT' 15
'~--' ''
```

Аутентификационный файл /etc/ppp/pap-secrets, используемый для подключения к провайдеру, должен содержать следующую строку:

```
login-name провайдер пароль
```

где *провайдер* — это содержимое переменной remotename в конфигурационном файле. Для активизации соединения необходимо ввести команду pppd call my-isp.

В следующем примере PPP-соединение устанавливается поверх существующего Internet-соединения, но на этот раз с помощью команды ssh организуется защищенный туннель через виртуальную частную сеть. Мы покажем как серверную, так и клиентскую конфигурации.

Вот серверный файл /etc/ppp/options:

```
noauth
logfile pppd.log
passive
silent
nodetach
```

Для каждого соединения создается также файл /etc/ppp/options.*терминал*, содержащий IP-адрес соединения:

```
локальный_IP-адрес:удаленный_IP-адрес
```

На сервере интерпретатор команд пользователя PPP устанавливается равным /usr/sbin/pppd; это позволит запускать серверный демон автоматически. Все ключи аутентификации должны быть заданы заранее с помощью утилиты ssh-agent, чтобы никакие пароли не запрашивались. На стороне клиента конфигурационный файл хранится в каталоге /etc/ppp/peers и назван по имени сервера — пусть это будет "my-work". Клиентский файл /etc/ppp/peers/my-work содержит следующие записи:

```
noauth
debug
logfile pppd.log
passive
silent
pty "ssh -t пользователь@удаленный_хост"
```

Чтобы зарегистрироваться через защищенное соединение, пользователю достаточно ввести pppd call my-work.

Наконец, приведем пример, в котором используется утилита wvdial без всяких диалоговых сценариев:

/etc/wvdial.conf:

```
[Dialer Defaults]
Phone = номер_телефона
Username = регистрационное_имя
Password = пароль
Modem = /dev/ttyS1
[Dialer creditcard]
Phone = междугородный_код,,,номер_телефона,,номер_услуги
```

Если утилита wvdial вызывается без аргументов, она руководствуется параметрами из секции Dialer Defaults файла /etc/wvdial.conf или ~/.wvdialrc для осуществления звонка и установки PPP-соединения. Будучи же вызванной с параметром (например, wvdial creditcard), утилита берет установки из соответствующей секции файла, переопределяя стандартные параметры.

Чтобы разорвать PPP-соединение, лучше воспользоваться командой `ifdown`, а не уничтожать демон `pppd`. Во втором случае Linux замечает факт уничтожения демона и перезапускает его.

```
$ sudo ifdown ppp0
```

Если система установлена на портативном компьютере, который иногда подключается к сети посредством Ethernet, а не PPP, на момент запуска демона `pppd` может существовать стандартный маршрут через интерфейс Ethernet. К сожалению, демон слишком "вежлив", чтобы удалить этот маршрут и установить свой собственный. Для исправления ситуации просто удалите "лишний" маршрут с помощью команды `ifdown`.

Вот как выглядят конфигурация PPP-интерфейса и таблица маршрутизации после установки соединения:

```
$ ifconfig ppp0
ppp0 Link encap:Point-to-Point Protocol
     inet addr:10.0.0.56   P-t-P:10.0.0.55   Mask:255.255.255.255
     UP POINTOPOINT RUNNING NOARP MULTICAST   MTU:1500   Metric:1
     RX packets:125 errors:0 dropped:0 overruns:0 frame:0
     TX packets:214 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:3
     RX bytes:11446 (11.1 Kb)   TX bytes:105586 (103.1 Kb)

$ netstat -nr
Kernel IP routing table
Destination   Gateway    Genmask           Flags  MSS  Window  irtt   Iface
10.0.0.55     0.0.0.0    255.255.255.255   UH     40   0       0      ppp0
0.0.0.0       10.0.0.55  0.0.0.0           UG     40   0       0      ppp0
```

Узнать статистику PPP-соединения позволяет команда `pppstats`:

```
$ pppstats
   IN   PACK   VJCOMP   VJUNC   VJERR |     OUT  PACK   VJCOMP   VJUNC   NON-VJ
11862   133         8      96       0 | 110446   226       27      89      110
```

В колонке `VJCOMP` показано значение счетчика TCP-пакетов, для которых применяется сжатие заголовков по методу Ван Якобсона. В колонке `VJUNC` подсчитано число несжатых пакетов. Подробности можно узнать в документе RFC1144.

Отладка PPP-соединения довольно затруднительна из-за большого числа участвующих сторон. Демон `pppd` регистрирует свои сообщения в системе Syslog от имени средства `daemon` (в Red Hat и Debian) либо от имени средства `local2` (в SUSE). Чтобы повысить уровень отладки, задайте в строке запуска демона флаг `debug` или измените установки конфигурационного файла. Кроме того, у демона предусмотрены различные коды завершения. Если запуск демона завершился неудачно, выполните команду `echo $status`, чтобы узнать код завершения, а затем поищите описание этого кода на `man`-странице демона.

В SUSE каждая подсистема снабжается образцами конфигурационных файлов, в комментариях к которым объясняются формат и назначение доступных параметров. Файлы в каталоге `/etc/ppp` — не исключение. Они хорошо документированы и предлагают вполне разумные значения многих параметров.

В Debian для протокола PPP также имеются хорошо документированные образцы конфигурационных файлов. Каталог `/etc/chatscripts` выделен для диалоговых сценариев. Чтобы сетевой интерфейс активизировался по протоколу PPP, включите его определение в файл `/etc/network/interfaces` (метод `ppp`), связав с помощью параметра

provider имя провайдера (в данном случае “my-isp”) с именем файла в каталоге /etc/ppp/peers (/etc/ppp/peers/my-isp). Например:

```
iface eth0 inet ppp
   provider my-isp
```

В Debian имеются специальные команды pon и poff, предназначенные для управления PPP-соединениями.

12.14. Особенности сетевого конфигурирования в Linux

В отличие от большинства систем, Linux обращает внимание на флаги поля TOS (Type of Service — тип обслуживания) IP-пакета и быстрее обслуживает пакеты, помеченные как интерактивные (минимальная задержка). К сожалению, ненормальное поведение систем компании Microsoft заставляет нас отказаться от нее.

Все пакеты, поступающие от более ранних систем Windows, помечаются как интерактивные независимо от их назначения. UNIX-системы этого не делают. Если Linux-шлюз обслуживает смешанную сеть, в которой есть как UNIX-, так и Windows-хосты, Windows-пакеты всегда будут иметь приоритет. В смешанной среде с устаревшими версиями операционных систем Windows снижение производительности для UNIX-хостов окажется ощутимым.

Отменить сортировку пакетов на основании поля TOS можно на этапе компиляции ядра Linux. Нужно просто отключить опцию “IP: use TOS value as routing key”.

Когда используется IP-маскирование (система NAT), ядро собирает полный пакет из фрагментов, прежде чем перенаправить его, даже если сразу после этого придется заново разбивать пакет на фрагменты. На дефрагментацию уходит большое количество циклов работы центрального процессора, однако современные процессоры настолько производительны, что подобная потеря времени вряд ли окажется заметной.

Linux позволяет менять MAC-адреса сетевых интерфейсов определенных типов:

```
redhat$ ifconfig eth1
eth1 Link encap:Ethernet  HWaddr 00:02:B3:19:C8:87
     BROADCAST MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:100
     Interrupt:7 Base address:0xee80

redhat$ sudo ifconfig eth1 hw ether 00:02:B3:19:C8:21

redhat$ ifconfig eth1
eth1 Link encap:Ethernet  HWaddr 00:02:B3:19:C8:21
     BROADCAST MULTICAST  MTU:1500  Metric:1
     RX packets:0 errors:0 dropped:0 overruns:0 frame:0
     TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:100
     Interrupt:7 Base address:0xee80
```

Это довольно опасная возможность, которая может привести к неправильной работе устройства. Пользоваться ею следует лишь в крайних случаях.

12.15. Рекомендуемая литература

- Stevens, W. Richard. *TCP/IP Illustrated, Volume One: The Protocols*. Reading, MA: Addison-Wesley, 1994.
- Wright, Gary R., and W. Richard Stevens. *TCP/IP Illustrated, Volume Two: The Implementation*. Reading, MA: Addison-Wesley, 1995.

  Эти два тома являются прекрасным, исчерпывающим руководством по протоколам TCP/IP, хотя уже немного устарели.

- Stevens, W. Richard. *UNIX Network Programming*. Upper Saddle River, NJ: Prentice Hall, 1990.
- Stevens, W. Richard, Bill Fenner, and Andrew M. Rudoff. *UNIX Network Programming, Volume 1, The Sockets Networking API (3rd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2003.
- Stevens, W. Richard. *UNIX Network Programming, Volume 2: Interprocess Communications (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 1999.

  Эти книги являются библиями для студентов, изучающих сетевое программирование. В первом издании описан только интерфейс сокетов Беркли. В третьем издании рассмотрены также интерфейс STREAMS и стандарт IPv6. Все три книги написаны в четком и понятном стиле Рича Стивенза.

- Tanenbaum, Andrew. *Computer Networks (4th Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2003.

  Эта книга по-прежнему является классическим учебником по сетям. Она содержит подробнейшее описание физического и канального уровней стека сетевых протоколов. Последнее ее издание включает в себя описание беспроводных сетей, высокоскоростных Ethernet-сетей, одноранговых сетей, передачи голосовых данных посредством IP и многого другого.

- Salus, Peter H. *Casting the Net, From ARPANET to INTERNET and Beyond*. Reading, MA: Addison-Wesley Professional, 1995.

  Это захватывающая история о том, как сеть ARPANET превратилась в Internet. Книга написана историком, который многие годы общался с профессионалами UNIX, поэтому кажется, будто ее писал один из них!

- Comer, Douglas. *Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architectures (5th Edition)*. Upper Saddle River, NJ: Pearson Prentice Hall, 2006. (Дуглас Камер. *Сети TCP/IP, том 1. Принципы, протоколы и структура. 4-е издание*. ИД "Вильямс", 2003)

  Книги серии *Internetworking with TCP/IP* Дугласа Камера долгое время считались эталонными справочниками по протоколам TCP/IP. Они предназначены в качестве учебного пособия для студентов и являются хорошим источником справочного материала.

- Hedrick, Charles. *Introduction to the Internet Protocols*. Rutgers University, 1987.

  Этот документ может служить введением в TCP/IP. У него нет постоянного “местожительства”, но он широко распространен в Internet.

- Hunt, Craig. *TCP/IP Network Administration (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2002.

  Эта книга ориентирована на администраторов UNIX-систем. Половина книги посвящена протоколам TCP/IP, а остальной материал касается средств UNIX более высокого уровня, таких как электронная почта и удаленный доступ.

Прекрасная коллекция документов, посвященных истории сети Internet и различных ее технологий, находится по адресу `www.isoc.org/internet/history`.

12.16. Упражнения

12.1. Почему прием директив переадресации протокола ICMP может привести к несанкционированному доступу в сеть?

12.2. Что такое значение MTU сетевого канала? Что произойдет, если это значение будет слишком большим? А слишком низким?

☆ 12.3. Объясните смысл концепции подсетей. Для чего нужны подсети? Что такое сетевая маска? Как сетевая маска связана с разделением IP-адреса на сетевую и машинную части?

☆ 12.4. Сеть 134.122.0.0/16 разбита на подсети с длиной маски 19.

а) Сколько будет подсетей /19? Перечислите их. Каковы их сетевые маски?

б) Сколько хостов может быть в каждой подсети?

в) Определите, какой подсети принадлежит адрес 134.122.67.124?

г) Каким будет широковещательный адрес каждой подсети?

☆ 12.5. С хоста 128.138.2.4 в сети 128.138.2.0/24 нужно послать пакет хосту 128.138.129.12 в сети 128.138.129.0/24. Предполагается, что:

- хост 128.138.2.4 имеет стандартный шлюз 128.138.2.1;
- хост 128.138.2.4 только что загрузился и еще не отправлял и не принимал пакеты;
- все остальные компьютеры сети работают уже долгое время;
- маршрутизатор 128.138.2.1 имеет прямое соединение с хостом 128.138.129.1, который является шлюзом в подсеть 128.138.129.0/24.

а) Перечислите все этапы отправки пакета. Покажите исходные и целевые MAC- и IP-адреса передаваемых пакетов.

б) А не изменился бы ответ, если бы сеть имела адрес 128.138.0.0/16? Аргументируйте свой ответ.

в) Если бы сеть 128.138.2.0 имела маску /26, а не /24, изменился бы ответ? Если да, то каким бы он был?

☆☆ 12.6. Как учесть рассмотренные в этой главе проблемы безопасности в только что инсталлированной Linux-системе? Проверьте, все ли проблемы решены в Linux-системах вашей компьютерной лаборатории. (Может требоваться доступ с правами суперпользователя.)

☆☆ 12.7. Какие действия необходимо предпринять для подключения нового компьютера к сети в вашей лаборатории? В ответе используйте параметры конкретной среды. Предполагается, что на новом компьютере уже установлена операционная система Linux.

☆☆ 12.8. Приведите текст конфигурационного файла DHCP-сервера, который выделяет адреса в диапазоне 128.138.192.[1-55]. Задайте срок аренды равным двум часам и сделайте так, чтобы хост с MAC-адресом 00:10:5A:C7:4B:89 всегда получал IP-адрес 128.138.192.55.

Глава 13

Маршрутизация

Управление потоками данных — непростая задача. В главе 12 вкратце рассказывалось о перенаправлении IP-пакетов. В настоящей главе мы подробнее изучим этот процесс и познакомимся с различными протоколами, благодаря которым маршрутизаторы автоматически находят наиболее эффективные маршруты. Эти протоколы принимают на себя основную нагрузку по управлению таблицами маршрутизации, существенно упрощая задачу администраторам. Они также позволяют быстро перенаправлять трафик в случае выхода из строя маршрутизатора или сетевого сегмента.

Важно отличать реальное перенаправление IP-пакетов от управления маршрутной таблицей, хотя в обоих случаях употребляют один и тот же термин — "маршрутизация". Перенаправление пакетов — простая процедура, тогда как вычисление маршрутов происходит по довольно запутанному алгоритму. Мы познакомимся только с одноадресной маршрутизацией, так как при многоадресной (групповой) маршрутизации возникает целый ряд новых проблем, рассмотреть которые, учитывая объем книги, не представляется возможным.

В подавляющем большинстве ситуаций информация, приведенная в главе 12 — это все, что нужно знать о маршрутизации. При наличии готовой сетевой инфраструктуры можно определить единственный статический маршрут (см. раздел "Маршрутизация" далее в главе), и этого будет вполне достаточно для получения доступа практически ко всем хостам в Internet. Если же нужно использовать сложную сетевую топологию или же система Linux должна применяться для управления частью сетевой инфраструктуры, вам может пригодиться информация о протоколах динамической маршрутизации и программных средствах, приведенная в данной главе.

Народная мудрость гласит, что IP-маршрутизация — чрезвычайно сложная тема, которая понятна только кучке длинноволосых хиппи, обитающих в подземельях университета Беркли. На самом деле это не так; достаточно лишь вспомнить, что в основе маршрутизации лежит принцип нахождения "следующего перехода". В любой конкретной точке требуется определить только следующий хост или маршрутизатор, куда будет отправлен пакет на пути к пункту назначения. Это в корне отличается от многих старых протоколов, где перед отправкой пакета в сеть требовалось задать точный его маршрут (принцип маршрутизации "от источника")[1].

13.1. Подробнее о маршрутизации пакетов

Прежде чем приступать к изучению алгоритмов маршрутизации, рассмотрим подробнее, как используются маршрутные таблицы. Предположим, сеть имеет топологию, изображенную на рис. 13.1.

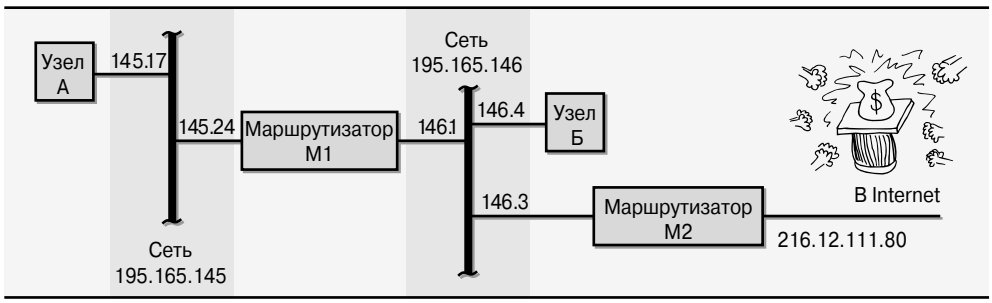


Рис. 13.1. Демонстрационная сеть

Маршрутизатор M1 соединяет два сетевых сегмента, а маршрутизатор M2 соединяет одну из сетей с внешним миром (считаем, что M1 и M2 — Linux-компьютеры, а не специализированные маршрутизаторы). Посмотрим, как выглядят таблицы маршрутизации. Вот таблица хоста A:

```
A$ netstat -rn
Kernel IP routing table
Destination     Gateway       Genmask       Flags  MSS  Window  irtt  Iface
199.165.145.0   0.0.0.0       255.255.255.0 U      0    0       0     eth0
127.0.0.0       0.0.0.0       255.0.0.0     U      0    0       0     lo
0.0.0.0         199.165.145.24 0.0.0.0      UG     0    0       0     eth0
```

Подробнее о команде `ifconfig` рассказывалось в разделе 12.7.

Хост A имеет самую простую конфигурацию среди всех четырех компьютеров. Первые два маршрута описывают собственные сетевые интерфейсы хоста. Они необходимы, чтобы пакеты, направляемые в непосредственно подключенные сети, не маршрутизировались особым образом. Устройство `eth0` — это Ethernet-плата хоста A, а `lo` — интерфейс обратной связи (виртуальный сетевой интерфейс, эмулируемый ядром).

[1] Маршрутизатор может работать с пакетами, в которых явно указан адрес следующего перехода, но это почти никогда не делается. Из соображений безопасности подобный режим чаще всего недоступен.

Обычно такие записи автоматически добавляются командой `ifconfig` при конфигурировании сетевого интерфейса.

Стандартный маршрут хоста А (третий в списке) задает перенаправление всех пакетов, не адресованных интерфейсу обратной связи или сети 195.165.145, на маршрутизатор М1, адрес которого в данной сети — 199.165.145.24. Флаг `G` указывает на то, что данный маршрут ведет к шлюзу, а не к одному из локальных интерфейсов хоста А. Шлюзы должны находиться на расстоянии одного перехода.

О различных типах адресации речь шла в разделе 12.3.

Предположим теперь, что хост А посылает пакет хосту Б с адресом 199.165.146.4.IP-подсистема ищет маршрут к сети 199.165.146 и, не найдя такового, отправляет пакет по стандартному маршруту, т.е. маршрутизатору М1. На рис. 13.2 приведена структура пакета, проходящего по сети Ethernet (в заголовке Ethernet указаны MAC-адреса соответствующих интерфейсов компьютеров А и М1 в сети 145).

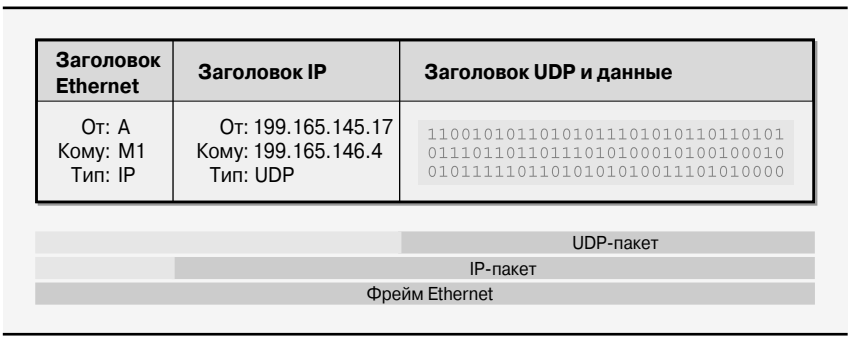


Рис. 13.2. Ethernet-пакет

Целевой аппаратный Ethernet-адрес соответствует маршрутизатору М1, но IP-пакет, скрытый в Ethernet-фрейме, не содержит никаких упоминаний о маршрутизаторе. Когда маршрутизатор просматривает поступивший пакет, он обнаруживает, что не является его получателем. Тогда он обращается к собственной таблице маршрутизации, чтобы узнать, как переслать пакет хосту Б, не переписывая его IP-заголовок (необходимо, чтобы отправителем пакета оставался хост А).

Так выглядит таблица маршрутизации хоста М1:

```
R1$ netstat -rn
Kernel IP routing table
Destination     Gateway       Genmask       Flags  MSS Window  irtt Iface
127.0.0.0       0.0.0.0       255.0.0.0     U      0   0       0    lo
199.165.145.0   0.0.0.0       255.255.255.0 U      0   0       0    eth0
199.165.146.0   0.0.0.0       255.255.255.0 U      0   0       0    eth1
0.0.0.0         199.165.146.3 0.0.0.0       UG     0   0       0    eth1
```

Она почти аналогична таблице хоста А, за исключением того, что здесь присутствуют два физических сетевых интерфейса. Стандартный маршрут в данном случае ведет к хосту М2, поскольку через него осуществляется выход в Internet. Пакеты, адресованные любой из сетей 199.165, доставляются напрямую.

Подобно хосту А, хост Б имеет один реальный сетевой интерфейс. Но для корректного функционирования ему необходим дополнительный маршрут, поскольку хост имеет пря-

мое соединение сразу с двумя маршрутизаторами. Трафик сети 199.165.145 должен проходить через хост М1, а весь остальной трафик — направляться в Internet через хост М2.

```
B$ netstat -rn
Kernel IP routing table
Destination     Gateway       Genmask       Flags MSS  Window  irtt  Iface
127.0.0.0       0.0.0.0       255.0.0.0     U     0    0       0     lo
199.165.145.0   199.165.146.1 255.255.255.0 U     0    0       0     eth0
199.165.146.0   0.0.0.0       255.255.255.0 U     0    0       0     eth0
0.0.0.0         199.165.146.3 0.0.0.0       UG    0    0       0     eth0
```

О директивах переадресации протокола ICMP рассказывалось в разделе 12.5.

Можно сконфигурировать хост Б так, чтобы изначально он “знал” только об одном шлюзе, полагаясь на помощь директив переадресации протокола ICMP для устранения избыточных переходов. Ниже показан пример начальной конфигурации:

```
B$ netstat -rn
Kernel IP routing table
Destination     Gateway       Genmask       Flags MSS  Window  irtt  Iface
127.0.0.0       0.0.0.0       255.0.0.0     U     0    0       0     lo
199.165.146.0   0.0.0.0       255.255.255.0 U     0    0       0     eth0
0.0.0.0         199.165.146.3 0.0.0.0       UG    0    0       0     eth0
```

Теперь, если хост Б посылает пакет хосту А (199.165.145.17), прямой маршрут найден не будет, и пакет отправится на хост М2. Поскольку хост М2 является маршрутизатором, он имеет полную информацию о состоянии сети и, следовательно, “знает” о роли хоста М1, куда и будет послан пакет. Кроме того, маршрутизатор М2 обнаружит, что хосты М1 и Б находятся в одной сети, поэтому он дополнительно направит хосту Б ICMP-сообщение, в соответствии с которым хост Б добавит в свою таблицу маршрутизации прямой маршрут к хосту А:

```
199.165.145.17  199.165.146.1  255.255.255.255  UGHD  0  0       0  eth0
```

Благодаря этому маршруту весь последующий трафик, адресованный хосту А, будет идти непосредственно через маршрутизатор М1. Однако это изменение не затрагивает трафик к другим хостам в сети, которой принадлежит хост А. Для них нужно получить отдельные директивы от маршрутизатора М2.

Некоторые администраторы выбирают для своих систем директивы переадресации протокола ICMP в качестве основного “протокола” маршрутизации, думая, что подобный подход обеспечит большую динамичность. К сожалению, если информация, касающаяся переадресованного маршрута, изменяется, нужно либо удалить маршрут из таблицы вручную, либо перезагрузить систему. Из-за этой и ряда других проблем (увеличение сетевого трафика, возрастание нагрузки на маршрутизатор М2, хаос в таблице маршрутизации, зависимость от внешних серверов) мы не рекомендуем применять такой подход в конфигурациях рассмотренного типа. В правильно сконфигурированной сети переадресованные маршруты не должны появляться в таблице маршрутизации.

13.2. Демоны и протоколы маршрутизации

В простейших сетях, подобно той, что представлена на рис. 13.1, имеет смысл настраивать маршрутизацию вручную. Но с определенного момента сети становятся слишком сложными для подобного администрирования. Вместо того чтобы явно сообщать

каждому компьютеру, как находить другие компьютеры и сети, лучше заставить сами компьютеры искать эту информацию. Данная задача возлагается на протоколы маршрутизации и демоны, которые их реализуют.

Протоколы маршрутизации имеют то преимущество перед системами статической маршрутизации, что они позволяют быстро адаптироваться к изменениям в топологии сети. Когда пропадает канал, демоны быстро находят альтернативные маршруты и сообщают о них в сети, связанные с этим каналом.

Демоны маршрутизации черпают информацию из трех источников: конфигурационных файлов, существующих таблиц маршрутизации и "родственных" демонов других систем. Собранные данные объединяются, и вычисляется оптимальный набор маршрутов, после чего новые маршруты записываются в системную таблицу (и при необходимости посылаются другим системам посредством протоколов маршрутизации). Состояние сети время от времени меняется, поэтому демоны должны периодически опрашивать друг друга, чтобы убедиться в актуальности имеющейся у них информации.

Конкретный алгоритм вычисления маршрутов зависит от протокола. Последние бывают двух типов: дистанционно-векторные и топологические.

Дистанционно-векторные протоколы

В основе дистанционно-векторных протоколов лежит следующая идея: если маршрутизатор X находится в пяти переходах от сети Y и является моим соседом, то я нахожусь в шести переходах от данной сети. Демон, работающий по такому протоколу, объявляет о том, как далеко, по его расчетам, расположены известные ему сети. Если соседние демоны не "знают" более коротких маршрутов к этим сетям, они помечают данный компьютер как оптимальный шлюз. В противном случае они просто игнорируют этот анонс[2]. Предполагается, что со временем таблицы маршрутизации придут в стабильное состояние.

Это довольно красивая идея, и если бы все работало так, как задумано, маршрутизация существенно упростилась бы. К сожалению, описанный алгоритм не лучшим образом справляется с изменениями топологии. Иногда стабилизация таблиц вообще не наступает вследствие возникновения бесконечных циклов (например, маршрутизатор X получает информацию от маршрутизатора Y и посылает ее маршрутизатору Z, который возвращает ее маршрутизатору Y). На практике приходится вводить сложные эвристические правила или задавать ограничения. К примеру, в RIP (Routing Information Protocol — протокол маршрутной информации) считается, что любая сеть, находящаяся на расстоянии более пятнадцати переходов, недоступна.

Даже в обычных ситуациях может потребоваться слишком много циклов обновлений, прежде чем все маршрутизаторы перейдут в стабильное состояние. Следовательно, чтобы не превысить допустимые пределы, необходимо сделать периоды обновлений очень короткими, а это, в свою очередь, ведет к тому, что дистанционно-векторные протоколы оказываются слишком "словоохотливыми". Например, протокол RIP требует, чтобы маршрутизаторы осуществляли широковещательную рассылку всей имеющейся у них информации каждые 30 секунд. В протоколах IGRP и EIGRP обновления анонсируются каждые 90 секунд.

[2] Вообще-то не все так просто, поскольку определенные изменения топологии могут приводить к удлинению существующих маршрутов. Некоторые дистанционно-векторные протоколы, например EIGRP, хранят информацию обо всех возможных маршрутах, чтобы на крайний случай был "план отступления". Впрочем, конкретные детали не так уж важны.

В противоположность этому в BGP (Border Gateway Protocol — протокол пограничного шлюза) вся таблица посылается один раз, после чего распространяются только изменения, когда они происходят. Такая оптимизация позволяет существенно снизить "переговорный" трафик (большей частью ненужный).

В табл. 13.1 перечислены дистанционно-векторные протоколы, широко используемые в настоящее время.

Таблица 13.1. Распространенные дистанционно-векторные протоколы маршрутизации

Аббревиатура	Полное название	Область применения
RIP	Routing Information Protocol	Локальные сети
IGRP	Interior Gateway Routing Protocol (протокол маршрутизации внутренних шлюзов) (не рекомендуется для использования)	Некоторые глобальные сети
EIGRP	Enhanced Interior Gateway Routing Protocol (улучшенный протокол маршрутизации внутреннего шлюза)	Глобальные сети, корпоративные локальные сети
BGP	Border Gateway Protocol (протокол пограничного шлюза)	Магистральные сети Internet

Топологические протоколы

В топологических протоколах, или протоколах состояния канала, информация рассылается в относительно необработанном виде. Записи выглядят примерно так: "Маршрутизатор X является смежным по отношению к маршрутизатору Y, и канал функционирует". Полный набор таких записей образует карту сетевых связей, на основании которой каждый маршрутизатор может сформировать собственную таблицу. Основное преимущество топологических протоколов по сравнению с дистанционно-векторными заключается в способности быстро стабилизировать таблицы маршрутизации в случае непредвиденного сбоя. К издержкам относится необходимость хранения полной карты соединений на каждом хосте, для чего требуются дополнительные процессорные мощности и память.

Поскольку процесс общения между маршрутизаторами не является частью собственно алгоритма вычисления маршрутов, появляется возможность реализовать топологические протоколы так, чтобы не возникало циклов. Изменения в топологической базе данных распространяются по сети очень быстро, не перегружая ни канал, ни центральный процессор.

Топологические протоколы сложнее дистанционно-векторных, зато они позволяют реализовать такие технологии, как маршрутизация на основании запрашиваемого типа обслуживания (поле TOS IP-пакета) и поддержка нескольких маршрутов к одному адресату. Ни одна из этих возможностей не реализована в готовых Linux-системах, и для их использования придется покупать специализированные маршрутизаторы.

В табл. 13.2 перечислены популярные топологические протоколы.

Таблица 13.2. Распространенные топологические протоколы маршрутизации

Аббревиатура	Полное название	Область применения
OSPF	Open Shortest Path First (открытый протокол первоочередного обнаружения кратчайших маршрутов)	Крупные сети
IS-IS	Intermediate System to Intermediate System (протокол маршрутизации между промежуточными системами)	Лабораторные экспериментальные сети, сети психиатрических лечебниц

Метрики стоимости

Чтобы протокол маршрутизации мог определить, какой путь к заданной сети является кратчайшим, необходимо, прежде всего, выяснить, что значит "кратчайший"[3]. Это путь с наименьшим числом переходов? С наименьшей задержкой? С наименьшими финансовыми затратами?

Для целей маршрутизации качество канала связи определяется числом, называемым метрикой стоимости. Стоимость пути — это сумма стоимостей составляющих его каналов связи. В простейших системах каждому каналу назначается стоимость 1, и в результате метрикой маршрута становится число переходов. Но любой из перечисленных выше критериев может являться метрикой стоимости.

Эксперты в области сетей долго и упорно трудились над тем, чтобы определение такого понятия, как метрика стоимости, было максимально гибким, а некоторые современные протоколы даже позволяют использовать разные метрики для разных видов сетевого трафика. Тем не менее, в 99% случаев на все это можно не обращать внимания. Для большинства систем вполне подойдут стандартные метрики.

Бывают ситуации, когда кратчайший физический маршрут к адресату не должен быть выбран по умолчанию из административных соображений. В таких случаях можно искусственно завысить метрики критических каналов. Всю остальную работу предоставьте демонам.

Внутренние и внешние протоколы

Автономная система — это группа сетей, находящихся под административным или политическим контролем одного юридического лица. Такое определение довольно расплывчато. Реальные автономные системы могут представлять собой как глобальные корпоративные сети, так и сети университетов или даже отдельных факультетов. Все зависит от того, как осуществляется маршрутизация. Сейчас наблюдается тенденция к укрупнению автономных систем. Это упрощает администрирование и повышает эффективность маршрутизации.

Маршрутизация внутри автономной системы отличается от маршрутизации между такими системами. Протоколы второго типа (внешние, или протоколы внешних шлюзов) должны управлять множеством маршрутов к различным сетям и учитывать тот факт, что соседние маршрутизаторы находятся под контролем других людей. Внешние протоколы не раскрывают топологию автономной системы, поэтому в определенном смысле их можно рассматривать как второй уровень маршрутизации, на котором соединяются группы сетей, а не отдельные компьютеры.

[3] Хорошо хоть не нужно выяснять, что означает слово "значит".

На практике небольшим и среднего размера организациям редко требуется внешний протокол, если только они не подключены к нескольким провайдерам одновременно. При наличии нескольких провайдеров традиционное разделение на локальный домен и домен Internet нарушается, поскольку маршрутизаторам приходится определять, какой маршрут в Internet лучше всего подходит для данного конкретного адреса. (Это не означает, что каждый маршрутизатор должен знать всю необходимую информацию. Большинство хостов может направлять свои пакеты внутреннему шлюзу, хранящему необходимые сведения.)

Внешние протоколы не сильно отличаются от своих внутренних аналогов, но именно о последних пойдет речь в данной главе, так как они применяются гораздо чаще.

Для получения дополнительных сведений о внешних протоколах обратитесь к разделу 13.8.

13.3. Основные протоколы маршрутизации

В этом разделе мы познакомимся с основными внутренними протоколами маршрутизации, узнаем их преимущества и недостатки.

RIP: протокол маршрутной информации

RIP — это старый протокол компании Xerox, адаптированный для IP-сетей (определен в документе RFC1058). Он используется демоном `routed` и представляет собой простой дистанционно-векторный протокол, метрикой стоимости в котором является число переходов. Поскольку протокол RIP разрабатывался в те времена, когда отдельные компьютеры стоили сотни тысяч долларов, а сети были относительно маленькими, в нем предполагается, что все хосты, находящиеся на расстоянии пятнадцати и более переходов, недоступны. По этой причине в крупных локальных сетях, где вдоль одного пути могут встретиться более пятнадцати маршрутизаторов, протокол RIP применять нельзя.

Несмотря на то что вследствие расточительного использования широковещательного режима протокол RIP стал истинным "пожирателем" ресурсов, он весьма эффективен при частых изменениях структуры сети, а также в том случае, когда топология удаленных сетей неизвестна. Однако после выхода из строя какого-нибудь канала стабилизация системы может занять довольно много времени.

Во многих системах демон `routed` запускается с опцией `-q`, означающей режим подавления сообщений. В этом режиме он управляет таблицей маршрутизации и принимает сообщения об изменении маршрутов, но не осуществляет широковещательную рассылку своей собственной информации. Вычислением маршрутов занимается более эффективный протокол, например OSPF. Найденные маршруты преобразуются в RIP-анонсы и рассылаются клиентским компьютерам. Демон `routed`, работающий в режиме подавления сообщений, поддерживается почти повсеместно, поэтому большинство систем может пользоваться преимуществами динамической маршрутизации без специального конфигурирования.

Протокол RIP широко используется не на Linux-платформах. Многие устройства, включая сетевые принтеры и SNMP-компоненты, способны принимать RIP-анонсы, узнавая о возможных шлюзах. Кроме того, демон `routed` присутствует во всех версиях UNIX и Linux, поэтому протокол RIP де-факто считается "наименьшим общим знаме-

нателем" среди протоколов маршрутизации. Как правило, он применяется в пределах локальной сети, тогда как глобальную маршрутизацию осуществляют демоны более мощных протоколов.

RIP–2: протокол маршрутной информации, версия 2

О протоколе CIDR рассказывалось в разделе 12.4.

RIP-2 — это улучшенная версия протокола RIP, включающая поддержку ряда особенностей, которые отсутствуют в исходном варианте протокола. Самое важное изменение заключается в том, что вместе с адресом следующего перехода передается сетевая маска. Это упрощает управление сетями, где есть подсети и применяется протокол CIDR. Была также предпринята попытка усилить безопасность протокола RIP, но определение схемы аутентификации оставлено для будущих разработок.

Протокол RIP-2 обладает рядом особенностей, ориентированных на использование в многопротокольной среде. Например, благодаря сообщениям об обновлении следующего перехода широковещательные станции могут анонсировать маршруты, для которых они не являются шлюзами. А посредством специальных маршрутных меток информация о маршрутах, обнаруженных внешними протоколами, может распространяться по протоколу RIP.

Демон протокола RIP-2 способен работать в режиме совместимости, сохраняя большинство новых особенностей протокола, но в то же время обслуживая и обычных RIP-клиентов. Протокол RIP-2 предпочтительнее, чем RIP, хотя в Linux он не поддерживается по умолчанию.

OSPF: открытый протокол первоочередного обнаружения кратчайших маршрутов

Протокол OSPF является топологическим. Его спецификация дана в документе RFC2328. Термин "первоочередное обнаружение кратчайших маршрутов" обозначает специальный математический алгоритм, по которому вычисляются маршруты; термин "открытый" — синоним слова "непатентованный".

OSPF был первым топологическим протоколом, нашедшим широкое применение. Он и до сих пор остается одним из самых популярных протоколов. Своей популярностью он обязан демону многопротокольной маршрутизации `gated`, о котором мы подробно поговорим ниже. К сожалению, протокол очень сложен, поэтому имеет смысл использовать его только в крупных системах (где важна эффективность маршрутизации).

В спецификации протокола OSPF не навязывается конкретная метрика стоимости. В реализации, разработанной компанией Cisco, по умолчанию в качестве метрики стоимости используется число переходов, но прокол можно сконфигурировать также для использования пропускной способности сети.

OSPF — протокол промышленного уровня, который эффективно функционирует в крупных сетях со сложной топологией. По сравнению с протоколом RIP он имеет ряд преимуществ, включая возможность управления несколькими маршрутами, ведущими к одному адресату, и возможность разделения сети на сегменты ("области"), которые будут делиться друг с другом только высокоуровневыми данными маршрутизации.

IGRP и EIGRP: протоколы маршрутизации внутренних шлюзов

IGRP и его преемник EIGRP — это патентованные протоколы, используемые только маршрутизаторами Cisco. Протокол IGRP был разработан для устранения некоторых недостатков протокола RIP еще в те времена, когда не было такого стабильного протокола, как OSPF. В настоящее время вместо него рекомендуется использовать протокол EIGRP, но он все еще применяется во многих организациях. Протокол EIGRP конфигурируется аналогично IGRP, хотя внутренне он реализован совсем по-другому. Протокол IGRP понимает только маршрутные анонсы, в которых IP-адреса заданы в соответствии с традиционными границами классов, тогда как EIGRP понимает произвольные сетевые маски CIDR.

Оба протокола являются дистанционно-векторными, но они спроектированы так, чтобы избежать проблем зацикливания и медленной стабилизации, свойственных другим протоколам данного класса. В этом смысле протокол EIGRP считается образцом. Для большинства применений протоколы EIGRP и OSPF обеспечивают равные функциональные возможности.

По нашему мнению, лучше придерживаться открытого, широко используемого протокола, каковым является OSPF. С ним многие работают, и существует несколько его общедоступных реализаций.

IS–IS: протокол маршрутизации между промежуточными системами

Протокол IS-IS является ответом на протокол OSPF со стороны организации ISO. Первоначально он предназначался для маршрутизации в рамках стека OSI, но впоследствии был расширен на стек TCP/IP.

Оба протокола — IS-IS и OSPF — создавались в начале 90-х гг., когда протоколы ISO были в моде и считались стандартом. Благодаря усилиям со стороны организации IETF протокол IS-IS нашел применение в сетях TCP/IP, но со временем стал все сильнее уступать в популярности протоколу OSPF и сегодня редко используется вне тестовых сред сертификации поставщиков. В силу множества ненужных функциональных особенностей, заложенных в него организацией ISO, лучше его избегать.

MOSPF, DVMRP и PIM: протоколы многоадресной маршрутизации

MOSPF (Multicast OSPF — многоадресный протокол OSPF), DVMRP (Distance Vector Multicast Routing Protocol — дистанционно-векторный протокол многоадресной маршрутизации) и PIM (Protocol Independent Multicast — протокольно-независимая многоадресная маршрутизация) предназначены для поддержки группового вещания в IP-сетях, хотя эта технология еще не нашла широкого применения. Информацию о протоколах можно найти на Web-хосте `www.mbone.com`.

Протокол обнаружения маршрутизаторов

Протокол обнаружения маршрутизаторов использует ICMP-сообщения, посылаемые по групповому IP-адресу 224.0.0.1, для распространения информации о других маршру-

тизаторах в сети. К сожалению, не все маршрутизаторы в настоящее время рассылают такие сообщения, и не все компьютеры могут их принимать. Остается надеяться, что когда-нибудь этот протокол станет более популярным.

13.4. Демон routed: стандартный демон маршрутизации

Демон `routed` распространен почти повсеместно. Долгое время он был стандартным демоном маршрутизации в UNIX, и его до сих пор включают в большинство дистрибутивов UNIX и Linux.

В Linux демон `routed` понимает только протокол RIP. Если планируете использовать протокол RIP-2, можно воспользоваться легко конфигурируемым демоном маршрутизации компании Nexus, доступным на сайте `sourceforge.net/projects/nx-routed`. Применение протокола RIP-2 важно, только если маски подсетей не совпадают с границами байта.

Демон `routed` способен работать в режиме сервера (`-s`) или в режиме подавления сообщений (`-q`). В обоих режимах принимаются широковещательные сообщения, но собственную информацию рассылают лишь серверы. Вообще говоря, серверами должны быть только устройства с несколькими сетевыми интерфейсами. Если не указана ни опция `-s`, ни `-q`, демон `routed` работает в режиме подавления сообщений при наличии одного интерфейса и в режиме сервера — при наличии нескольких.

О команде `route` рассказывалось в разделе 12.7.

Демон заносит обнаруженные маршруты в таблицу ядра. Все маршруты должны анонсироваться как минимум каждые четыре минуты, иначе они будут удалены. Правда, это не относится к статическим маршрутам, добавляемым с помощью команды `route`.

Для отладки подсистемы маршрутизации можно воспользоваться командой `routed -t`. Опция `-t` заставляет демон перейти в интерактивный режим и отображать все посланные и полученные пакеты.

Демон `routed`, как правило, выявляет маршрутную информацию динамически и не требует конфигурирования. Если же в системе есть шлюзы в Internet или в другие автономные системы, то, вероятно, понадобятся дополнительные усилия для того, чтобы заставить демон `routed` работать с этими каналами.

Когда имеется всего один выходной шлюз, можно объявить его стандартным глобальным шлюзом, запустив на нем демон `routed` с флагом `-g`. Это аналогично заданию стандартного маршрута на одиночном компьютере, но в данном случае установка распространяется на всю сеть.

С демоном `routed` также связан конфигурационный файл `/etc/gateways`, к которому, по первоначальному замыслу, демон должен был обращаться при запуске для получения статической информации о шлюзах.

13.5. Демон gated: открытие неведомых возможностей

Демон `gated` явился фантастической и, на то время, бесплатной программой маршрутизации, позволяющей одновременно использовать различные протоколы. Он позво-

ляет точно управлять анонсированными маршрутами, а также широковещательными адресами, правилами безопасности и метриками стоимости. Он разрешает нескольким протоколам использовать одни и те же маршруты, позволяя тем самым создавать шлюзы между автономными областями, в которых приняты разные системы маршрутизации. Наконец, демон `gated` обладал одним из самых удобных командных интерфейсов среди всего административного ПО в Linux; это относится и к формату его конфигурационного файла.

Изначально демон `gated` распространялся бесплатно, но в 1992 г. он был приватизирован и перешел под контроль консорциума Merit GateD Consortium. Коммерческие версии демона доступны лишь членам упомянутого консорциума. Впоследствии консорциум прекратил свое существование, а права на демон `gated` были приобретены компанией NextHop, специализирующейся на разработке встроенного сетевого программного обеспечения. По сути, это привело к прекращению существования версий демона `gated` с открытым исходным кодом.

Брешь, образовавшуюся с уходом демона `gated`, призван заполнить новый многообещающий проект XORP (eXtensible Open Router Platform — расширяемая платформа маршрутизатора с открытым исходным кодом). Хотя в настоящее время XORP проходит этап бета-тестирования, он практически готов к промышленному применению, и можно надеяться, что он получит достаточное развитие, чтобы заполнить нишу, ранее занимаемую демоном `gated`. Последнюю информацию о состоянии разработки этой программы можно получить на сайте `www.xorp.org`.

А пока в качестве многопротокольного маршрутизатора можно использовать пакет маршрутизатора Zebra, разрабатываемого в рамках проекта GNU (`www.zebra.org`), который может работать на большинстве платформ Linux. К сожалению, ему недостает многих функциональных возможностей, средств облегчения использования и подробной документации, необходимых для управления динамической маршрутизацией в производственной среде. Вероятно, это один из тех случаев, когда приобретение специализированного маршрутизатора (подобного выпускаемым компаниями Juniper или Cisco) — наиболее рациональный способ решения задачи.

13.6. Выбор стратегии маршрутизации

Существуют четыре уровня сложности, характеризующих процесс управления маршрутизацией в сети:

- отсутствие маршрутизации как таковой;
- только статическая маршрутизация;
- преимущественно статическая маршрутизация, но клиенты принимают RIP-обновления;
- динамическая маршрутизация.

Общая топология сети существенно влияет на маршрутизацию каждого конкретного сегмента. В различных сетях могут требоваться совершенно разные уровни поддержки маршрутизации. При выборе стратегии руководствуйтесь перечисленными ниже эмпирическими правилами.

- Автономная сеть не нуждается в маршрутизации.

- Если из сети есть только один выход во внешний мир, клиенты сети (нешлюзовые хосты) должны иметь стандартный статический маршрут к этому шлюзу. Никакой другой конфигурации не требуется, кроме как на самом шлюзе.
- Можно задать явный статический маршрут к шлюзу, ведущему в небольшую группу сетей, и стандартный маршрут к шлюзу, ведущему во внешний мир. Но динамическая маршрутизация предпочтительнее, если до нужной сети можно добраться разными маршрутами.
- Если используется протокол RIP, старайтесь не запускать демон `routed` в режиме сервера, так как он передает все, что "знает" (независимо от того, верны эти сведения или нет), в широковещательном режиме через короткие промежутки времени, вследствие чего возникает избыточная нагрузка на сеть и систему. Для того чтобы клиенты пассивно ожидали поступления обновленных маршрутов и не рассылали собственную информацию, указывайте при запуске демона `routed` опцию `-q`.
- Многие считают, что RIP — кошмарный протокол, а демон `routed` — настоящее исчадие ада. Это клевета! Если демон работает в системе и обеспечивает нормальную производительность, пользуйтесь им на здоровье. Не стоит тратить время на разработку сложных схем маршрутизации.
- Демон `routed` принимает данные отовсюду, не проверяя их. Даже если в сети используется протокол RIP, можно управлять обменом маршрутными данными с помощью демона `gated`, а демон `routed` запускать только на клиентских компьютерах.
- Динамическую маршрутизацию следует применять в тех точках, где сети пересекают политические или административные границы.
- В сетях с динамической маршрутизацией, содержащих замкнутые контуры или избыточные пути, желательно применять протокол OSPF.
- Маршрутизаторы, подключенные к Internet-магистрали, должны использовать протокол BGP. Обычно подключение осуществляется в одной точке, поэтому достаточно задать простой стандартный статический маршрут.

Для организации среднего размера с относительно стабильной локальной структурой, где есть выход в другую сеть, подойдет сочетание статической и динамической маршрутизаций. Компьютеры локальной сети, которые не являются шлюзами во внешние сети, могут использовать статическую маршрутизацию, направляя все неизвестные пакеты стандартному шлюзу, способному общаться с внешним миром и выполнять динамическую маршрутизацию.

Сеть, управлять которой по такой схеме было бы слишком сложно, должна основываться на динамической маршрутизации. В краевых сетях по-прежнему можно использовать стандартные статические маршруты, но компьютеры в сетях с более чем одним маршрутизатором должны запускать демон `routed` в режиме подавления сообщений.

13.7. Маршрутизаторы Cisco

Маршрутизаторы, выпускаемые компанией Cisco Systems, Inc., сегодня являются стандартом де-факто на рынке маршрутизаторов. Захватив более 70% рынка, компания Cisco добилась широкой известности своих продуктов, к тому же имеется множество специалистов, умеющих работать с этими устройствами. Раньше в качестве маршрутиза-

торов часто приходилось использовать UNIX-системы с несколькими сетевыми интерфейсами. Сегодня специализированные маршрутизаторы стоят в коммутационных шкафах и над подвесными потолками, где сходятся сетевые кабели. Эти маршрутизаторы дешевле, быстрее и безопаснее, чем их UNIX- или Linux-аналоги.

Большинство маршрутизаторов Cisco работает под управлением операционной системы IOS, которая является собственностью компании Cisco и не имеет отношения к Linux. У нее достаточно большой набор команд: полная бумажная документация занимает на полке почти полтора метра. Мы никогда не смогли бы полностью описать здесь эту операционную систему, но все же постараемся дать о ней основные сведения.

В IOS определены два уровня доступа (пользовательский и привилегированный), каждый из которых включает механизм парольной защиты. По умолчанию к маршрутизатору Cisco можно получить доступ на пользовательском уровне, воспользовавшись утилитой telnet[4]. Будет выдано приглашение на ввод пароля:

```
$ telnet acme-gw.acme.com
Connected to acme-gw.acme.com.
Escape character is '^]'.

User Access Verification
Password:
```

После ввода правильного пароля появится приглашение интерпретатора EXEC:

```
acme-gw.acme.com>
```

С этого момента можно начинать вводить команды, например `show interfaces` для отображения списка сетевых интерфейсов маршрутизатора или `show ?` для получения справки о других командах.

Чтобы перейти в привилегированный режим, введите команду `enable`, а при последующем запросе — пароль. Признаком привилегированного режима является наличие символа "#" в конце строки приглашения:

```
acme-gw.acme.com#
```

Будьте осторожны! В данном режиме можно делать все что угодно, включая удаление конфигурационной информации и даже самой операционной системы.

Команда `show running` позволяет узнать текущие динамические параметры маршрутизатора, а по команде `show config` отображаются текущие неизменяемые параметры. В большинстве случаев оба набора данных идентичны.

Вот типичная конфигурация:

```
acme-gw.acme.com# show running
Current configuration:
version 12.1
hostname acme-gw
enable secret xxxxxxxx
ip subnet-zero

interface Ethernet0
description Acme internal network
ip address 192.108.21.254 255.255.255.0
no ip directed-broadcast
```

[4] Можно сконфигурировать множество методов доступа. Если в вашей организации уже используются маршрутизаторы Cisco, обратитесь к сетевому администратору, чтобы узнать, какие методы доступа разрешены.

```
interface Ethernet1
description Acme backbone network
ip address 192.225.33.254 255.255.255.0
no ip directed-broadcast

ip classless
line con 0
transport input none

line aux 0
transport input telnet
line vty 0 4
password xxxxxxxx
login

end
```

Модифицировать конфигурацию маршрутизатора можно разными способами. Компания Cisco предлагает графические утилиты, работающие в некоторых версиях UNIX/Linux и Windows, но опытные сетевые администраторы никогда ими не пользуются. Их самый мощный "инструмент" — командная строка. Кроме того, с помощью команды `tftp` можно загрузить с маршрутизатора его конфигурационный файл и отредактировать в любимом текстовом редакторе, после чего снова записать файл в память маршрутизатора[5].

Чтобы отредактировать конфигурационный файл в режиме командной строки, введите команду `config term`:

```
acme-gw.acme.com# config term
Enter configuration commands, one per line.   End with CNTL/Z.
acme-gw(config)#
```

Теперь можно вводить конфигурационные команды в том виде, в котором их будет отображать команда `show running`. Например, если требуется изменить IP-адрес интерфейса `Ethernet0`, введите:

```
interface Ethernet0
ip address 192.225.40.253 255.255.255.0
```

По окончании ввода конфигурационных команд нажмите <Ctrl+Z>, чтобы вернуться к обычной командной строке. Если все прошло успешно, сохраните новую конфигурацию в постоянной памяти посредством команды `write mem`.

Приведем несколько советов, касающихся эффективной работы с маршрутизаторами Cisco.

- Присвойте маршрутизатору имя с помощью команды `hostname`. Это позволит избежать несчастных случаев, связанных с изменением конфигурации не того маршрутизатора. Заданное имя всегда будет отображаться в командной строке.
- Всегда храните резервную копию конфигурационного файла маршрутизатора. Можно написать короткий сценарий на языке Expect, который будет с помощью команды `tftp` каждую ночь пересылать текущую конфигурацию в Linux-систему.

[5] Полезный совет: Microsoft Word — не лучший выбор в данном случае.

- Контролируйте доступ к командной строке маршрутизатора, создавая списки доступа для каждого порта VTY (аналогичен порту PTY в Linux). Это не позволит посторонним пользователям "взломать" маршрутизатор[6].
- Управляйте трафиком между сетями (по возможности и во внешний мир тоже), создавая списки доступа для каждого интерфейса. Для получения более подробной информации о настройке списков доступа обратитесь к разделу 20.12, посвященному брандмауэрам.
- Физически ограничивайте доступ к маршрутизаторам. Ведь если у злоумышленника будет физический доступ к устройству, он легко сможет изменить пароль привилегированного режима.

13.8. Рекомендуемая литература

- Perlman, Radia. *Interconnections: Bridges, Routers, Switches, and Internetworking Protocols (2nd Edition)*. Reading, MA: Addison-Wesley, 2000.

  Эта книга содержит исчерпывающую информацию по данной теме. Если вы желаете приобрести только одну книгу по основам организации сетей, эта работа — именно то, что требуется. Кроме того, не пренебрегайте шансом пообщаться с Радья — она очень остроумна и хранит в памяти поразительный объем сведений.

- Huitema, Christian. *Routing in the Internet (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR, 2000.

  Эта книга представляет собой отличное введение в маршрутизацию для начинающих. В ней описано большинство используемых сегодня протоколов маршрутизации, а также рассматривается ряд сложных тем, например групповое вещание.

- Moy, John T. *OSPF: Anatomy of an Internet Routing Protocol*. Reading, MA: Addison-Wesley, 1998.

  Эта книга содержит исчерпывающее описание протокола OSPF, сделанное автором самого протокола.

- Stewart, John W *BGP4 Inter-domain Routing in the Internet*. Reading, MA: Addison-Wesley, 1999.

Имеется также множество документов RFC, посвященных маршрутизации. Основные из них перечислены в табл. 13.3.

Таблица 13.3. Документы RFC, посвященные маршрутизации

RFC	Название	Авторы
2328	OSPF Version 2	John T. Moy
1058	Routing Information Protocol	C. Hedrick
2453	RIP Version 2	Gary Scott Malkin
1256	ICMP Router Discovery Messages	Stephen E. Deering

[6] В последних версиях операционной системы IOS поддерживается протокол SSH, поэтому старайтесь пользоваться командой `ssh`, а не `telnet`.

Окончание табл. 13.3

RFC	Название	Авторы
1142	OSI IS-IS Intra-domain Routing Protocol	David R. Oran
1075	Distance Vector Multicast Routing Protocol	D. Waitzman и др.
4632	CIDR: an Address Assignment and Aggregation Strategy	Vince Fuller и др.
4271	A Border Gateway Protocol 4 (BGP-4)	Yakov Rekhter и др.

13.9. Упражнения

13.1. Изучите возможности команды `route` в Linux и составьте короткое описание ее работы. Как с помощью этой команды

а) добавить стандартный маршрут к шлюзу 128.138.129.1 через интерфейс `eth1`?

б) удалить маршрут 128.138.129.1?

в) определить, каким образом был добавлен маршрут: с помощью демона, например `gated`, или директивы переадресации протокола ICMP (ту же информацию можно получить, анализируя вывод команды `netstat -rn`)?

13.2. Сравните статический и динамический способы маршрутизации, укажите достоинства и недостатки каждого способа. Опишите ситуации, для которых подходит тот или иной вид маршрутизации, и объясните почему.

☆ 13.3. Команда `netstat -rn` сообщает показанные ниже результаты. Опишите каждый маршрут и нарисуйте общую схему сети. Какая сеть — 10.0.0.0 или 10.1.1.0 — "ближе" к Internet? Как появился каждый из маршрутов?

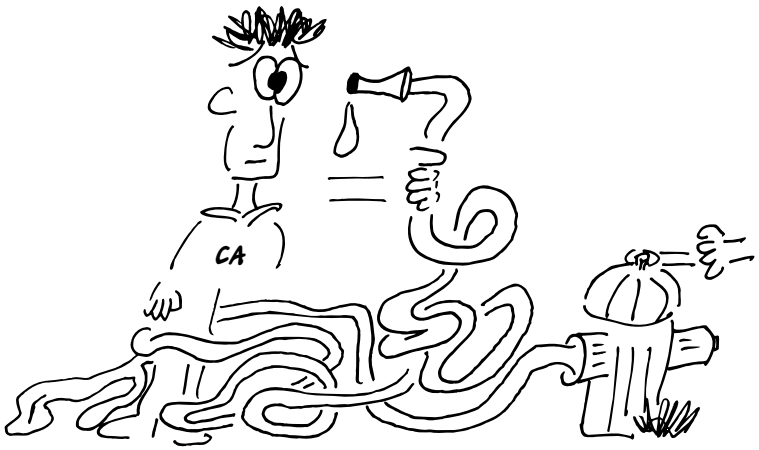
```
Destination   Gateway    Genmask         Flags  MSS   Window   irtt  Iface
10.0.0.0      0.0.0.0    255.255.255.0   U      40    0        0     eth1
10.1.1.0      0.0.0.0    255.255.255.0   U      40    0        0     eth0
0.0.0.0       10.0.0.1   0.0.0.0         UG     40    0        0     eth1
```

☆☆ 13.4. Определите стратегию маршрутизации в вашей системе. Какие протоколы используются? Какие компьютеры подключены непосредственно к сети Internet? Воспользуйтесь утилитой `tcpdump` для просмотра маршрутных обновлений, курсирующих в локальной сети, и командой `traceroute` для выхода за пределы локальной сети. (Необходим доступ с правами суперпользователя.)

☆☆ 13.5. Если бы вы были провайдером среднего размера, предоставляющим услуги коммутируемого доступа и виртуального хостинга, какую конфигурацию системы маршрутизации вы бы использовали? Убедитесь в том, что учитываются не только шлюзы между Internet-магистралью и вашей сетью, но и любые внутренние маршрутизаторы. Нарисуйте схему маршрутизации в сети.

Глава 14

Сетевые аппаратные средства



В современном мире практически все виды информации могут храниться и обрабатываться в цифровой форме. При этом все заинтересованы в максимально быстрой передаче данных из одного места в другое. За всем этим стоит самое разное сетевое оборудование и обслуживающее его программное обеспечение, многие компоненты которого берут свое начало в UNIX. Если и существует область, в которой технология UNIX оказывает непосредственное влияние на жизнь людей, то это ее практическая реализация передачи больших объемов пакетных данных.

Актуальная на сегодня задача — угнаться за стремительно развивающимся миром. Конечно, быстродействие и надежность сетей непосредственно влияют на результаты деятельности компаний, но в настоящее время сети столь вездесущи, что их состояние может оказывать влияние на возможность осуществлять многие обычные действия, связанные с общением — например, выполнять телефонные вызовы. Недостатки в построении сетей приводят к различным личным и профессиональным проблемам, которые могут иметь буквально катастрофические социальные последствия. А устранение этих недостатков может быть весьма дорогостоящим.

Успешное создание сети зависит от четырех важнейших факторов, а именно:

- разработки приемлемой структуры сети;
- выбора высококачественного оборудования;
- правильной инсталляции и документирования;
- квалифицированного выполнения операций по эксплуатации и обслуживанию.

В этой главе рассматриваются среды передачи данных, которые чаще всего используются для организации локальных и глобальных сетей, в том числе Ethernet, ATM, переключение фреймов, беспроводные сети и DSL. Кроме того, освещаются вопросы проектирования, которые актуальны для любых сетей — как новых, так и существующих.

14.1. ЛВС, ГВС и ОВС

В определенном смысле нам повезло, что протоколы TCP/IP способны легко работать в самых разнообразных средах. Но в действительности рынок сетевых аппаратных средств не унифицирован, а разбит на множество сегментов.

Сети, проложенные в пределах здания или группы зданий, называются локальными вычислительными сетями, или ЛВС. В них преобладают высокоскоростные дешевые линии передач. В глобальных вычислительных сетях (ГВС) конечные точки соединения географически могут быть удалены друг от друга на значительные расстояния (тысячи километров). В таких сетях высокоскоростные каналы стоят очень дорого, зато в сеть могут объединяться компьютеры, находящиеся в разных городах и странах. Общегородская вычислительная сеть (ОВС) — новый термин, обозначающий высокоскоростную умеренной стоимости среду, функционирующую в рамках города или группы городов-спутников. В этой главе мы узнаем, какие технологии стоят за всеми этими сетями.

14.2. Ethernet: общепризнанная ЛВС

Захватив более 90% мирового рынка ЛВС, сети Ethernet стали почти повсеместным явлением в современном компьютерном мире. Разработку стандарта Ethernet начал Боб Меткаф (Bob Metcalfe) из Массачусетсского технологического института в рамках своей кандидатской диссертации. После окончания учебы Боб стал работать в научно-исследовательском центре компании XEROX в Пало-Альто. Объединив усилия с корпорациями DEC и Intel, компания XEROX в конце концов реализовала стандарт Ethernet на коммерческом уровне. Этот проект стал одним из первых примеров сотрудничества конкурирующих компьютерных фирм[1].

В первоначальной спецификации Ethernet была определена скорость передачи данных 3 Мбит/с (она почти сразу же выросла до 10 Мбит/с). Всеобщее внимание технология привлекла в 1994 г., когда в соответствии с новой спецификацией скорость ее работы составила 100 Мбит/с. В 1998 г. был достигнут новый рубеж: 1 Гбит/с. Сегодня стандарт обеспечивает передачу данных по волоконным кабелям со скоростью равной 10 Гбит/с, оставляя позади всех своих конкурентов. Стандарт Ethernet для медного кабеля, обеспечивающий скорость передачи равную 10 Гбит/с (802.3an), был принят IEEE в июле 2006 г. В табл. 14.1 перечислены основные этапы эволюции различных стандартов Ethernet[2].

[1] Боб Меткаф сформулировал закон (“закон Меткафа”), который гласит, что ценность сети растет экспоненциально по мере увеличения числа пользователей.

[2] Мы не упомянули несколько стандартов, которым не удалось завоевать популярность, в частности 100BaseT4 и 100BaseVG-AnyLAN.

Таблица 14.1. Эволюция Ethernet

Год	Скорость	Название стандарта	IEEE№	Расстояние	Среда передачи[a]
1973	3 Мбит/с	Xerox Ethernet	–	?	Коаксиальный кабель
1976	10 Мбит/с	Ethernet 1	–	500 м	Коаксиальный кабель RG-11
1982	10 Мбит/с	DIX Ethernet (Ethernet II)	–	500 м	Коаксиальный кабель RG-11
1985	10 Мбит/с	10Base5 (“Thicknet”)	802.3	500 м	Коаксиальный кабель RG-11
1985	10 Мбит/с	10Base2 (“Thinnet”)	802.3	180 м	Коаксиальный кабель RG-58
1989	10 Мбит/с	10BaseT	802.3	100 m	Медный кабель НВП категории 3
1993	10 Мбит/с	10BaseF	802.3	2 km	ММ оптоволокно
				25 km	ОМ оптоволокно
1994	100 Мбит/с	100BaseTX (“100 meg”)	802.3u	100 m	Медный кабель НВП категории 5
1994	100 Мбит/с	100BaseFX	802.3u	2 km	ММ оптоволокно
				20 km	ОМ оптоволокно
1998	1 Гбит/с	1000BaseSX	802.3z	260 м	ММ оптоволокно (62,5 мкм)
				550 м	ММ оптоволокно (50 мкм)
1998	1 Гбит/с	1000BaseLX	802.3z	440 м	ММ оптоволокно (62,5 мкм)
				550 м	ММ оптоволокно (50 мкм)
				3 км	ОМ оптоволокно
1998	1 Гбит/с	1000BaseCX	802.3z	25 м	Двухпроводный экранированный кабель
1999	1 Гбит/с	1000BaseT (“Gigabit”)	802.3ab	100 м	Медный кабель НВП категорий 5E и 6
2002	10 Гбит/с	10GBase-SR	802.3ae	300 м	ММ оптоволокно
		10GBase-LR		10 км	ОМ оптоволокно
2006	10 Гбит/с	10GBase-T	802.3an	100 м	Медный кабель НВП категории 7
2006[б]	100 Гбит/с	TBD	TBD	TBD	Оптоволокно
2008[б]	1 Тбит/с	TBD	TBD	TBD	МГДДВ оптоволокно
2010[б]	10 Тбит/с	TBD	TBD	TBD	МТДДВ оптоволокно

[a] ММ — многомодовое, ОМ — одномодовое, НВП — неэкранированная витая пара, МГДДВ — мультиплексирование с грубым делением по длине волны, МТДДВ — мультиплексирование с точным делением по длине волны.

[б] Промышленная перспектива.

Как работает Ethernet

Технологию Ethernet можно представить в виде великосветского раута (рис. 14.1), на котором гости (компьютеры) не перебивают друг друга, а ждут паузы в разговоре (отсутствия трафика в сетевом кабеле), чтобы заговорить. Если два гостя начинают говорить одновременно (т.е. возникает конфликт), оба они останавливаются, извиняются друг перед другом, ждут немного, а затем один из них начинает говорить снова.

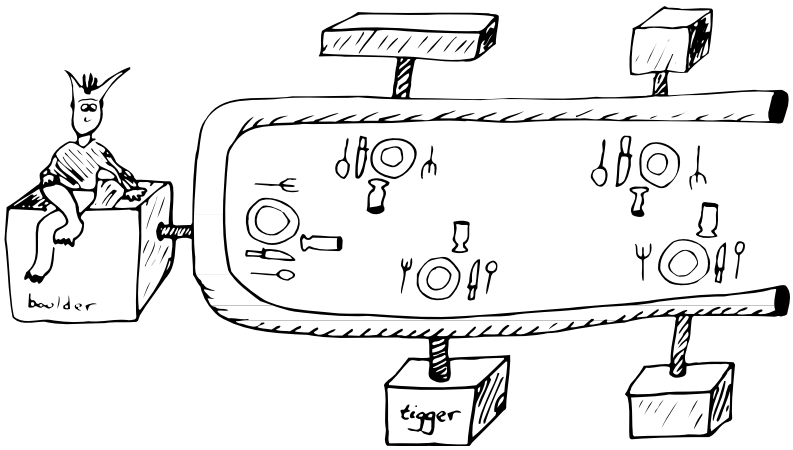


Рис. 14.1. "Великосветский раут" в сети Ethernet

В технической терминологии такая схема называется CSMA/CD (Carrier Sense Multiple Access with Collision Detection — множественный доступ с контролем несущей и обнаружением конфликтов). Вот смысл этого названия:

- контроль несущей — можно определить, занят ли канал;
- множественный доступ — кто угодно может передавать сообщения;
- обнаружение конфликтов — передающая система "знает", когда она "перебивает" кого-нибудь.

Фактическая задержка при обнаружении конфликтов является случайной величиной. Это позволяет избежать такого развития событий, при котором два компьютера одновременно передают сообщения в сеть, обнаруживают коллизию, ждут некоторое время, а затем синхронно возобновляют передачу, переполняя, таким образом, сеть конфликтами. Но так было не всегда!

Топология Ethernet

С точки зрения топологии сеть Ethernet представляет собой шину с ответвлениями, но без контуров. В Ethernet могут передаваться пакеты трех типов: однонаправленные, групповые и широковещательные. Пакеты первого типа адресованы одному узлу, второго — группе узлов, третьего — всем узлам сегмента.

Широковещательный домен — это совокупность узлов, которые принимают пакеты, направляемые по аппаратному широковещательному адресу. В каждом логическом Ethernet-сегменте существует ровно один широковещательный домен. В ранних стан-

дартах Ethernet (например, 10Base5) понятия физического и логического сегментов были тождественными, поскольку все пакеты передавались по одному большому кабелю, в который втыкались сетевые интерфейсы компьютеров[3].

С появлением коммутаторов логические сегменты стали включать в себя множество (десятки и даже сотни) физических (или, в ряде случаев, беспроводных) сегментов, к которым подключено всего два устройства: порт коммутатора и компьютер. Коммутаторы отвечают за доставку групповых и однонаправленных пакетов в физический (или беспроводный) сегмент, где расположен нужный адресат (адресаты); широковещательные пакеты направляются во все сетевые порты логического сегмента.

Логический сегмент может состоять из физических (или беспроводных) сегментов, имеющих разную скорость передачи данных (10 Мбит/с, 100 Мбит/с, 1 Гбит/с или 10 Гбит/с). Следовательно, коммутаторы должны иметь средства буферизации и синхронизации для предотвращения возможных конфликтов.

Неэкранированная витая пара

Неэкранированная витая пара (НВП) — самая популярная среда передачи данных в сетях Ethernet. Стандарт основан на звездообразной топологии и обладает рядом преимуществ по сравнению с другими средами:

- применяется недорогой, недефицитный медный кабель (иногда можно использовать готовую телефонную проводку);
- соединение на основе НВП гораздо проще смонтировать и наладить, чем соединение на основе коаксиального кабеля или оптоволокна, к тому же легко подобрать нужную длину кабеля;
- используются дешевые, надежные и удобные в эксплуатации разъемы RJ-45;
- все соединения функционируют независимо друг от друга, поэтому дефект кабельной системы в одном соединении не повлияет на другие компьютеры сети.

Общая схема сети на основе НВП изображена на рис. 14.2.

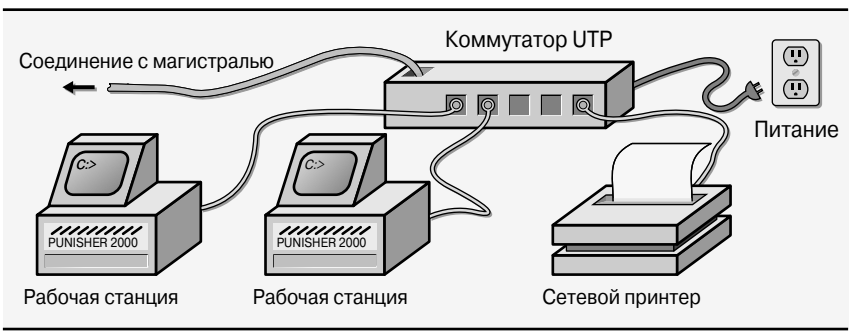


Рис. 14.2. Схема сети на основе НВП

[3] Мы не шутим! Подключение нового компьютера к сети предполагало прокалывание отверстия в изоляции кабеля с помощью специального соединителя, называемого “зуб вампира”, который позволял добраться до центрального проводника. Этот соединитель затем зажимался винтами.

Провода, используемые в современных ЛВС на основе НВП, обычно подразделяют на восемь категорий. Эта система оценки параметров была впервые введена компанией Anixter, крупным поставщиком кабельной продукции, и впоследствии стандартизирована организацией TIA (Telecommunications Industry Association — ассоциация телекоммуникационной промышленности). Сегодня выделяются категории 1–7 и промежуточная категория 5E.

Организация ISO тоже подключилась к процессу стандартизации кабелей и предложила собственную классификацию, которая почти в точности повторяет классификацию TIA. Например, кабель категории 5 в системе TIA эквивалентен кабелю класса D в системе ISO. Для наглядности в табл. 14.2 подытожены ключевые различия между основными современными стандартами кабелей.

Таблица 14.2. Характеристики кабелей НВП

Параметр[а]	Категория 5 Класс D[б]	Категория 5E	Категория 6 Класс E	Категория 7 Класс F
Ширина полосы	100 МГц	100 МГц	250 МГц	600 МГц
Затухание	24 дБ	24 дБ	21,7 дБ	20,8 дБ
NEXT	27,1 дБ	30,1 дБ	39,9 дБ	62,1 дБ
Разность затуханий	3,1 дБ	6,1 дБ	18,2 дБ	41,3 дБ
ELFEXT	17 дБ	17,4 дБ	23,2 дБ	?[в]
Затухание отраженного сигнала (обратная потеря)	8 дБ	10 дБ	12 дБ	14,1 дБ
Задержка распространения сигнала	548 нс	548 нс	548 нс	504 нс

[а] NEXT — Near-end crosstalk — ослабление перекрестной наводки на ближнем конце, ELFEXT — Equal level far-end crosstalk — ослабление равноуровневой перекрестной наводки на дальнем конце.

[б] Включая дополнительные спецификации TIA TSB 95 и ISO FDAM 2.

[в] Этот параметр в настоящее время не определен в ожидании дальнейших исследований.

Кабели категорий 1 и 2 годятся только для передачи звуковых сигналов. Кабель категории 3 является стандартом для самых медленных сетей: 10BaseT со скоростью передачи 10 Мбит/с. Кабель категории 4 ни для чего конкретного не предназначен. Иногда его применяют в сетях Token Ring на основе НВП со скоростью передачи 16 Мбит/с, а также в специализированных системах 10BaseT. Кабель категории 5 поддерживает работу на скорости 100 Мбит/с. Именно он чаще всего используется в современных кабельных системах. Кабели категорий 5E и 6 поддерживают скорость передачи 1 Гбит/с. Категория 7 зарезервирована для медных кабелей Ethernet со скоростью передачи 10 Гбит/с, стандарт которых еще только разрабатывается.

🕮 Подробнее о прокладке кабелей рассказывается в разделе 14.11.

Для соединений 10BaseT требуются две пары проводов категории 3, причем длина каждой линии передачи ограничена 100 метрами. В соединениях 100BaseTX предельная длина та же, но используются две пары проводов категории 5. Существуют провода с поливинилхлоридной и тефлоновой изоляцией. Выбор изоляции диктуется средой, в которой будут проложены кабели. В замкнутых помещениях, связанных с вентиляци-

онной системой здания, обычно требуется тефлоновая изоляция[4]. Поливинилхлоридная изоляция дешевле и проще в эксплуатации.

При монтаже соединений используются разъемы RJ-45, в которых задействованы выводы 1, 2, 3 и 6. Хотя для нормальной работы соединения со скоростью 10 или 100 Мбит/с достаточно двух пар проводов категории 3, в случае прокладки новой сети рекомендуется использовать кабель категории 5E с четырьмя парами проводов и подключать все восемь контактов разъема RJ-45.

Стандарт RS-232 описан в разделе 27.1.

Заводя четырехпарный НВП-кабель в коммутационные панели и настенные розетки RJ-45, придерживайтесь стандарта разводки TIA/EIA-568A. Этот стандарт, совместимый с другими вариантами RJ-45 (например, RS-232), позволяет избежать ошибок при разводке концов кабеля независимо от того, есть ли свободный доступ к парам. Требования стандарта отражены в табл. 14.3.

Таблица 14.3. Стандарт TIA/EIA-568A для подключения четырехпарного НВП-кабеля к розетке RJ-45

Пара	Цвета	Контакты разъема	Пара	Цвета	Контакты разъема
1	Белый/Синий	5/4	3	Белый/Зеленый	1/2
2	Белый/Оранжевый	3/6	4	Белый/Коричневый	7/8

Имеющаяся в здании проводка может подходить или не подходить для прокладки сетей, в зависимости от того, как и когда она прокладывалась.

Соединение и расширение сетей Ethernet

Согласно семиуровневой модели OSI, сети Ethernet можно логически соединять в нескольких точках. На первом уровне, физическом, используются разъемные соединения либо повторители (в наши дни заменяемые концентраторами). Они передают непосредственно электрические сигналы.

На втором уровне, канальном, применяются коммутаторы. Они передают сетевые фреймы, проверяя аппаратные адреса отправителя и получателя.

На третьем уровне, сетевом, используются маршрутизаторы. Они доставляют сообщение следующему узлу в соответствии с адресом получателя.

Концентраторы

Концентратор — активное устройство, используемое для соединения физических сегментов сетей Ethernet на основе НВП. Ему требуется внешний источник питания. Выступая в качестве повторителя, концентратор ретранслирует Ethernet-фреймы, но никак не интерпретирует их. Он “не имеет представления” ни о том, куда направляются пакеты, ни о том, какой протокол используется.

Две самые дальние точки сети должны отстоять друг от друга на расстояние не более четырех концентраторов. В первом и втором стандартах Ethernet число последователь-

[4] Конкретную информацию можно получить у пожарного инспектора или ответственного за пожарную безопасность.

но устанавливаемых концентраторов ограничивалось двумя на сеть. В спецификации IEEE 802.3 (10 Мбит/с) это число возросло до четырех. В сетях со скоростью передачи 100 Мбит/с допускаются два концентратора, в стандарте 1000BaseT — только один, а в сетях со скоростью передачи 10 Гбит/с — ни одного. На рис. 14.3 показаны допустимая и недопустимая конфигурации для десятимегабитной сети.

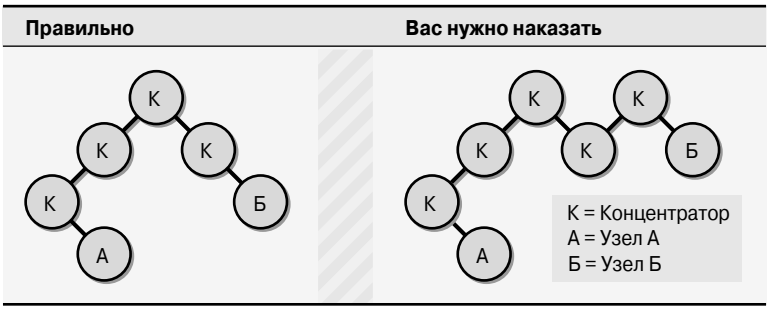


Рис. 14.3. Считаем концентраторы

Концентраторы время от времени требуют внимания со стороны системного администратора, поэтому их не следует размещать в труднодоступных местах. Включение и выключение питания обычно позволяет "оживить" устройство в случае зависания.

Коммутаторы

Коммутатор соединяет сети Ethernet на канальном (втором) уровне модели ISO. Его назначение — объединить две физические сети так, чтобы они представляли собой как бы одну большую физическую сеть. В настоящее время коммутаторы — стандартные устройства подключения устройств к сети Ethernet.

Коммутаторы принимают, регенерируют и повторно передают пакеты на аппаратном уровне[5]. В большинстве коммутаторов применяется алгоритм динамического обучения. Они запоминают, какие исходные адреса поступают с одного порта, а какие — с другого. Пакет передается из одного порта в другой только в том случае, если его получатель находится в соседнем сегменте. Первоначально пересылаются все пакеты, но через несколько секунд, когда коммутатор изучит расположение большинства узлов сети, запускается механизм фильтрации.

Поскольку между сетями пересылаются не все пакеты, каждый сегмент кабеля менее загружен, чем в случае, когда все компьютеры подключены к одному кабелю. А если учесть, что основной трафик имеет тенденцию к локализации, то увеличение реальной пропускной способности может оказаться довольно серьезным. Кроме того, коммутатор не влияет на логическую модель сети, поэтому его установка требует лишь незначительного вмешательства со стороны администратора.

Если сеть имеет замкнутые контуры, коммутатор может безнадежно запутаться, потому что пакеты, посылаемые одним компьютером, окажутся сразу на двух (или нескольких) портах коммутатора. В одной сети Ethernet замкнутых контуров не бывает, но

[5] Поскольку пакеты регенерируются и повторно синхронизируются, сети, сегменты которых соединены коммутаторами, лишены ограничения, подобного "счетчику концентраторов" (см. рис. 15.3).

после объединения нескольких таких сетей маршрутизаторами и коммутаторами топология изменится, вследствие чего может образоваться несколько путей к одному компьютеру. Ряд коммутаторов решает эту проблему путем резервирования альтернативных маршрутов на тот случай, если основной маршрут станет недоступным. Они упрощают топологию видимой ими сети, отсекая дублирующиеся пути до тех пор, пока в оставшихся сегментах не окажется только по одному маршруту к каждому узлу сети. Другие коммутаторы создают между сетями двойные каналы и переключают трафик по циклическому принципу.

Постоянное наращивание функциональных возможностей микропрограммного кода коммутаторов способствует повышению их "уровня интеллекта". Некоторые коммутаторы можно использовать для контроля безопасности сети. Они регистрируют все "чужие" Ethernet-адреса, выявляя таким образом новые ответвления на кабеле и сообщая о них. Поскольку коммутаторы функционируют на канальном уровне, они не зависят от протоколов и могут обрабатывать любые сочетания пакетов, использующих протоколы более высокого уровня (например, IP, AppleTalk или NetBEUI).

Коммутаторы просматривают каждый пакет, определяя, нужно ли его пересылать в другой сегмент. Производительность этих устройств обычно измеряют как скоростью просмотра пакетов, так и скоростью их пересылки. Многие поставщики не указывают в диаграммах производительности коммутаторов размеры тестировавшихся пакетов, поэтому реальная производительность может быть ниже объявленной.

Несмотря на то что быстродействие коммутаторов все время растет, эффективно использовать их можно при объединении в один логический сегмент не более сотни компьютеров. В крупных коммутируемых сетях часто возникают проблемы вроде "широковещательных штормов", поскольку широковещательный трафик должен проходить через все порты. Для решения этой проблемы нужно изолировать широковещательный трафик между коммутируемыми сегментами посредством маршрутизатора (создавая тем самым более одного логического Ethernet-сегмента).

В крупных организациях можно использовать коммутаторы, позволяющие разбивать их порты (программным путем) на группы, называемые виртуальными ЛВС (ВЛВС). ВЛВС — это группа портов, принадлежащая к одному логическому сегменту, как если бы порты были соединены со своим собственным выделенным коммутатором. Подобное секционирование позволяет повысить степень изоляции трафика, что полезно с точки зрения как безопасности, так и производительности.

Трафиком между виртуальными ЛВС управляет маршрутизатор или, в некоторых случаях, модуль маршрутизации самого коммутатора. Расширение этой системы, называемое "транкингом ВЛВС" (один из примеров реализации — протокол IEEE 802.1Q), позволяет разным коммутаторам обслуживать порты одной логической ВЛВС.

Выбор коммутатора может представлять определенную трудность. В этом сегменте рынка очень высокая конкуренция, следствием которой являются многочисленные рекламные заявления, не всегда подтверждаемые на практике. Поэтому не стоит особо доверять данным, которые приводятся поставщиками; лучше прислушаться к советам независимых экспертов (просмотрите тесты, приводимые в журналах). В последние годы нередко случалось так, что чей-то продукт оказывался "лучшим" в течение нескольких месяцев, а затем, после попыток внесения улучшений, его производительность или надежность падала ниже критической отметки.

В любом случае убедитесь, что скорость объединительной панели коммутатора является достаточной. У хорошо спроектированного коммутатора эта скорость должна превышать сумму скоростей всех его портов.

Маршрутизаторы

Маршрутизатор — это специализированное вычислительное устройство с двумя или несколькими сетевыми интерфейсами, работающее на третьем (сетевом) уровне модели ISO. Маршрутизаторы доставляют пакеты адресатам на основании информации, хранящейся в IP-заголовках. Помимо простого перемещения пакетов маршрутизаторы могут также выполнять ряд особых функций, например фильтрацию пакетов (в соответствии с правилами безопасности), разделение трафика по приоритетам (в соответствии с заданным качеством обслуживания) и обнаружение общей сетевой топологии. Подробно об алгоритмах маршрутизации рассказывалось в главе 13.

На одном маршрутизаторе могут быть установлены аппаратные интерфейсы разных типов (SONET, Ethernet, ATM). На программном уровне некоторые маршрутизаторы способны обрабатывать трафик, не относящийся к протоколу IP, например пакеты протоколов IPX и AppleTalk. В этом случае маршрутизатор и его интерфейсы необходимо сконфигурировать для каждого конкретного протокола. В настоящее время лучше отказаться от поддержки запатентованных протоколов и ограничиться надежной поддержкой только стека протоколов TCP/IP.

Конфигурация маршрутизаторов бывает фиксированной или модульной. Устройства первого типа содержат сетевые интерфейсы, установленные в заводских условиях. Они обычно подходят для специализированных применений. Например, маршрутизатор с интерфейсами T1 и Ethernet может оказаться удобным, когда нужно подключить небольшую компанию к сети Internet.

Модульные маршрутизаторы имеют слотовую или шинную архитектуру, а интерфейсы к ним добавляются пользователями. Как правило, это более дорогие устройства, но зато они обеспечивают большую гибкость в эксплуатации.

В зависимости от необходимой надежности и ожидаемого трафика специализированный маршрутизатор может оказаться как дороже, так и дешевле Linux-системы, сконфигурированной в качестве маршрутизатора. Однако специализированное устройство, как правило, обеспечивает более высокую производительность и надежность. Это та область сетевого проектирования, в которой лучше заранее вложить чуть больше денег, чем потом решать многочисленные проблемы.

14.3. Беспроводные сети: ЛВС путешественников

Беспроводные сети — стремительно развивающаяся область. Уже выпускаются продукты промышленного уровня по вполне приемлемой цене. К сожалению, в сравнении с успехами традиционных кабельных технологий скорость беспроводных сетей кажется несколько неадекватной для корпоративного применения, колеблясь в диапазоне от 2 до 54 Мбит/с. Впрочем, для многих применений этого вполне достаточно. Беспроводная сеть со скоростью 11 Мбит/с в условиях дома или небольшого офиса — мечта системного администратора. А беспроводная сеть со скоростью 54 Мбит/с годится для применения и в корпоративной среде. Беспроводная технология очень удобна для применения на выставках, в аэропортах и других многолюдных местах.

На сегодняшний день наиболее перспективными стандартами беспроводных сетей представляются IEEE-спецификации 802.11g и 802.11a. Сети по стандарту 802.11g работают в частотном диапазоне 2,4 ГГц и обеспечивают ЛВС-подобные подключения со скоростями до 54 Мбит/с. Дальность связи составляет от 100 м до 40 км в зависимости от используемого оборудования и условий местности. Стандарт 802.11a также обеспечи-

вает пропускную способность до 54 Мбит/с, но частотном диапазоне 5,4 ГГц. Некоторые современные устройства позволяют объединять два канала, обеспечивая пропускную способность равную 108 Мбит/с. Стандарт 802.11n обеспечивает пропускную способность более 200 Мбит/с.

Хотя в обоих названных стандартах заявлена пропускная способность равная 54 Мбит/с, реально она может быть значительно ниже. Стандарт 802.11g предназначен в первую очередь для потребительского рынка. Как правило, оборудование, изготовленное в соответствии с этим стандартом, дешевле изготовленного в соответствии со стандартом 802.11a и поддерживает три неперекрывающихся канала передачи данных, а не двенадцать, как по стандарту 802.11a. Эти каналы подобны полосам скоростной автомагистрали: чем больше каналов, тем большее число клиентом может в полной мере использовать полную пропускную способность линии.

Для небольших организаций вполне подойдет любой из названных стандартов. Администраторам больших организаций или университетских городков стоит остановить свой выбор на стандарте 802.11a, поскольку он предоставляет больше каналов связи. Большинство современных беспроводных радиоустройств может подключаться к сети любого типа.

Сегодня достаточно часто можно встретить сети 802.11b (11 Мбит/с), 802.11g и 802.11a. Соответствующие сетевые платы недороги и доступны для большинства портативных и настольных ПК (или же встроены в них). Как и в случае кабельных сетей Ethernet, в наиболее распространенной топологии сети 802.11 используется концентратор (называемый точкой доступа), к которому подключается множество клиентов. Точки доступа могут соединяться с традиционными кабельными сетями или друг с другом (беспроводным способом).

Чтобы сконфигурировать Linux-систему на работу в качестве точки доступа 802.11a/b/g, необходимы соответствующее оборудование и драйвер. Нам известен, по крайней мере, один специализированный набор микросхем — Intersil Prism II. Существует прекрасная специализированная беспроводная базовая станция AirPort Express компании Apple, предназначенная для домашнего и офисного применения. Она относительно недорога (около 150 долларов) и довольно функциональна[6].

Десятки фирм наперебой предлагают свои устройства беспроводного доступа. Их можно приобрести в специализированных и универсальных магазинах. Естественно, всегда следует помнить о том, что “вы получаете то, за что платите”. Дешевые устройства беспроводного доступа (цена которых не превышает 50 долларов) скорее всего, будут плохо справляться с передачей больших файлов или с обслуживанием нескольких активных клиентов.

Отладка беспроводной сети сродни черной магии. При решении возникающих при этом проблем приходится учитывать множество факторов. При развертывании беспроводной сети на уровне предприятия, вероятно, придется потратиться на приобретение анализатора беспроводной сети. Мы настоятельно рекомендуем оборудование для анализа, выпускаемое компанией AirMagnet.

Безопасность беспроводных сетей

Безопасность беспроводных сетей довольно невысока. Существует протокол WEP (Wired Equivalent Privacy — приватность, эквивалентная проводным сетям), применяе-

[6] Кроме всего прочего, она позволяет подключать к настольному или портативному ПК стереосистему.

мый в сетях 802.11b и обеспечивающий 40-, 104- либо 128-битное шифрование пакетов, передаваемых по радиоканалам. К сожалению, этот стандарт содержит фатальную проектную недоработку, которая превращает его всего лишь в досадную небольшую помеху на пути злоумышленников. Посторонний человек, находящийся за пределами здания, может получить прямой доступ в сеть и остаться незамеченным.

Появившийся недавно набор стандартов безопасности Wi-Fi Protected Access (защищенный доступ к Wi-Fi-сетям) (WPA) восстановил доверие к средствам обеспечения безопасности беспроводных сетей. Этот стандарт следует использовать во всех новых беспроводных сетях вместо стандарта WEP. Беспроводные сети, в которых не применяется стандарт WPA (независимо от наличия протокола WEP), следует считать полностью незащищенными.

802.11i или, иначе, WPA2 — более новая альтернатива стандарту WPA. Он содержит больше механизмов аутентификации, предназначенных для использования в беспроводных сетях предприятий.

Беспроводные коммутаторы

Подобно тому как концентраторы Ethernet развились до Ethernet-коммутаторов, оборудование беспроводных сетей постепенно совершенствуется для применения в средах больших предприятий. Ряд изготовителей (таких как Airespace) выпускает "беспроводные коммутаторы", которые работают в сочетании с множеством точек доступа, установленных в различных местах университетского городка. Идея состоит в том, чтобы установить множество недорогих точек доступа, которые затем будут управляться "интеллектуальным" коммутатором. Коммутатор хранит информацию конфигурирования протоколов WAP и обеспечивает беспрепятственную поддержку аутентификации и роуминга.

Если необходимо обеспечить повсеместную службу беспроводной связи в средних и больших организациях, безусловно, следует обратить свое внимание на эту категорию продуктов. Они не только уменьшают время, затрачиваемое на управление, но чаще всего предоставляют также средства слежения и управления качеством обслуживания, предоставляемого пользователям.

В качестве особенно эффективного подхода можно рекомендовать развертывание в организации сети 802.11a/b/g и ее использование для поддержки ручных устройств IP-телефонии, находящихся в распоряжении персонала. По затратам это сопоставимо с оплатой услуг сотовой связи!

14.4. FDDI: дорогая и разочаровывающая ЛВС

При скорости передачи 10 Мбит/с технология Ethernet 80-х гг. не обеспечивала пропускную способность, необходимую для решения некоторых сетевых задач, например соединения рабочих групп через корпоративную или университетскую магистраль. С целью создания более производительной среды комитет X3T9.5 организации ANSI (American National Standards Institute — Национальный институт стандартизации США) в середине 80-х гг. опубликовал альтернативный стандарт FDDI (Fiber Distributed Data Interface — распределенный интерфейс передачи данных по волоконно-оптическим каналам)[7]. Он разрабатывался и выпускался на рынок как интерфейс для сетей Token Ring со скоростью передачи 7 Мбит/с. Ожидалось, что технология FDDI позволит решить проблему пропускной способности, стоявшую перед многими организациями.

[7] FDDI был также принят в качестве стандарта ISO.

К сожалению, очень скоро стандарт всех разочаровал, и в настоящее время любое его промышленное применение совершенно неоправданно. Мы включили информацию о нем только ради полноты исторической картины (и на тот случай, если все еще используется в том или ином сегменте конкретной сети.)

📖 Подробнее о параметре MTU рассказывалось в разделе 12.3.

Для нормального качества функционирования FDDI-плате нужно гораздо большее значение MTU (максимальный размер передаваемого блока), нежели то, что по умолчанию установлено для Ethernet. Наилучший выбор — 4352 (задается с помощью команды `ifconfig`).

Стандарт FDDI определяет двухкольцевую топологию сети с передачей маркера, построенную на базе волоконно-оптической среды со скоростью передачи 100 Мбит/с (рис. 14.4). Здесь используются два кольца: основное обеспечивает передачу данных, а вспомогательное служит как резерв на случай разрыва.

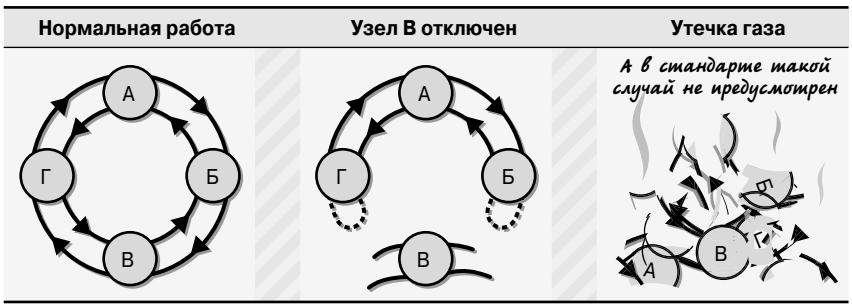


Рис. 14.4. Двойная кольцевая сеть FDDI

Компьютеры могут подключаться к обоим кольцам (в этом случае они называются узлами класса А, или узлами с двойным подключением) либо только к основному кольцу (узлы класса Б, с одинарным подключением). Магистральные маршрутизаторы и концентраторы чаще всего имеют двойное подключение, а рабочие станции — одинарное, через оптоволоконный концентратор.

Одно из преимуществ кольцевой сети с передачей маркера заключается в том, что доступом к такой сети управляет детерминированный протокол. Конфликтов доступа не возникает, поэтому производительность сети при высокой нагрузке не ухудшается, как это бывает в Ethernet. Многие кольцевые системы с маркерным доступом при обслуживании нескольких клиентов способны выходить на 90–95% своей номинальной пропускной способности.

Для физической среды стандарт FDDI предлагает два типа оптоволокна: одномодовое и многомодовое. Моды — это пучки световых лучей, входящих в оптоволокно под определенным углом. По одномодовому волокну может распространяться световая волна только одной частоты, поэтому в качестве источника излучения используется лазер[8].

[8] Большинство (но не все) лазеры, используемые в оптоволоконных сетях, относятся к устройствам класса 1 — т.е. их излучение может быть как “опасным”, так и безопасным для глаз, но само устройство сконструировано так, чтобы предотвратить излучение света наружу во время нормального использования. К сожалению, “нормальное использование” не предусматривает случай повреждения кабелей и поэтому в действительности гарантия безопасности отсутствует. Не светите лазером в глаза, даже если все окружающие делают это.

Многомодовое волокно допускает распространение света несколькими путями и обычно применяется с менее дорогостоящими и менее опасными светодиодами. Одномодовое волокно обеспечивает передачу информации на более дальние расстояния, чем многомодовое. Для FDDI чаще всего выбирают многомодовое волокно диаметром 62,5 микрона.

В FDDI применяются самые разные оптические разъемные соединения. Но независимо от того, какие разъемы используются, помните, что их чистота играет большую роль в обеспечении надежного функционирования волоконно-оптических линий связи. На рынке предлагаются специальные комплекты, предназначенные для самостоятельного монтажа соединений, однако лучше обратиться за помощью к профессионалам.

14.5. ATM: локальная сеть несбывшихся надежд

Аббревиатура ATM расшифровывается как Asynchronous Transfer Mode (асинхронный режим передачи), хотя некоторые настаивают, что она должна обозначать Another Technical Mistake (еще одна техническая ошибка). Один обозреватель охарактеризовал ATM как "попытку телефонной компании превратить работу в сети в некую услугу, за которую можно назначить тариф".

С технической точки зрения ATM является особой технологией, так как здесь проповедуется идея о том, что небольшие пакеты фиксированного размера (называемые *ячейками*) — самый эффективный способ реализации гигабитных сетей. Кроме того, ATM обещает возможности, которыми не располагают другие среды передачи, включая резервирование полосы пропускания и гарантированное качество обслуживания.

Размер ячейки в ATM составляет 53 байта. Для транспортировки ячеек определены пять так называемых уровней адаптации ATM (ATM Adaptation Layer, AAL). Их назначение описано в табл. 14.4.

Таблица 14.4. Уровни адаптации ATM

AAL	Применение
1	Передача с постоянной скоростью, например, голосовых данных (требуется минимальная задержка)
2	Передача с переменной скоростью, требующая лимитированной задержки сигнала
3	Передача данных с установлением соединений
4	Передача данных без установления соединений
5	Общие средства передачи данных (особенно IP-трафик); заменяет уровни 3 и 4

Не ясно, как на практике может использоваться уровень 2. Никакого стандарта для этого уровня пока нет. Уровни 3 и 4 оказались очень похожими и были объединены. Производители, которым пришлось реализовывать стандарт ATM, были весьма недовольны этими уровнями и разработали свое решение — уровень SEAL (Simple and Efficient Adaptation Layer — простой и эффективный уровень адаптации), который вскоре стал уровнем AAL 5.

ATM широко рекламировалась как универсальная коммутируемая среда для локальных, глобальных и общегородских вычислительных сетей. Но сегодня эта технология практически мертва, сохранившись лишь в глобальных сетях, с помощью которых круп-

ные телекоммуникационные корпорации пытаются вернуть инвестиции, недальновидно вложенные в ATM-оборудование.

Изготовители ATM-коммутаторов продолжают активно продвигать свои продукты и это оборудование можно заказать на множестве складов. Однако при развертывании новых сетей лучше использовать другие технологии.

14.6. Frame Relay: глобальная убыточная сеть

Ретрансляция кадров (frame relay) — это технология глобальных сетей, основанная на передаче данных с коммутацией пакетов, как правило, по приемлемым ценам. Часто говорят, что ретрансляция кадров представляет собой вновь выпущенный на рынок протокол X.25, применявшийся для доступа к сетям с коммутацией пакетов с середины 70-х гг. К счастью, протокол X.25 распространен достаточно широко, и за многие годы производители научились создавать оборудование и программное обеспечение, работающее надежно и эффективно.

В традиционной схеме пользователи, заинтересованные в установлении соединений с удаленными системами, должны были покупать у телефонной компании выделенный канал, например линию DDS (Digital Data Service — служба цифровой передачи данных) со скоростью 56 Кбит/с или линию T1. Это двухточечные каналы передачи данных, которые функционируют 24 часа в сутки. Увы, такое соединение стоит довольно дорого, так как телефонная компания должна выделять для канала оборудование и полосу частот.

В противоположность традиционной схеме технология ретрансляции кадров предполагает использование эффекта масштаба. Телефонная компания создает сеть (часто называемую "облаком"[9]), соединяющую между собой ее центральные станции (АТС). Пользователи посылают предназначенные для удаленных узлов данные в виде небольших пакетов. Телефонная компания коммутирует пакеты через соответствующие АТС и обеспечивает их доставку в пункты назначения. В такой модели пользователь и телефонная компания исходят из того, что в каждый момент времени трафик не превысит пропускной способности сети.

За доставку IP-трафика в сетях с ретрансляцией кадров отвечают маршрутизаторы. Пакеты коммутируются через невидимые постоянные виртуальные каналы (Permanent Virtual Channel, PVC), обеспечивающие получение пакетов только в тех пунктах, доставка в которые была оплачена. При этом гарантируется определенная степень конфиденциальности, что защищает информацию пользователя от получения ее другими абонентами, подключенными к той же сети.

Самое большое преимущество технологии ретрансляции кадров заключается в том, что она, как правило, недорога. В реальной жизни, однако, пользователь получает столько, сколько заплатил, и часто обнаруживает, что производительность оказывается недопустимо низкой. Доля накладных расходов на коммутацию пакетов в сетях с ретрансляцией кадров достаточно высока, поэтому в периоды значительной загруженности быстродействие канала может ухудшаться.

[9] Это весьма красноречивый термин, так как суть происходящего в сети с ретрансляцией кадров всегда довольно туманна.

14.7. ISDN: глобальная сеть-невидимка

ISDN (Integrated Services Digital Network — цифровая сеть с интегрированными услугами) представляет собой сервис телефонных компаний, способный принимать самые разнообразные формы. Наиболее распространен сервис BRI (Basic Rate Interface — интерфейс базового уровня). Это цифровая телефонная линия, которая включает два коммутируемых канала-носителя (называются B-каналами) со скоростью передачи 64 Кбит/с и один дополнительный конфигурационный дельта-канал (называемый D-каналом) со скоростью передачи 16 Кбит/с. Каждый из B-каналов можно использовать для доставки звука либо данных (звук можно передавать по одному каналу).

ISDN позволяет получить относительно быстродействующую цифровую линию связи по разумной цене (от 30 до 150 долларов в месяц в зависимости от того, где вы живете). Устройства, называемые терминальными адаптерами, превращают эту телефонную линию в более привычный интерфейс, например в RS-232. Они используются почти так же, как модемы, и стоят приблизительно столько же. Большинство адаптеров способно объединять два канала-носителя, формируя в результате один канал передачи данных со скоростью 128 Кбит/с.

ISDN может служить заменой обычным коммутируемым сетям, а также использоваться как глобальная технология, в которой посредством маршрутизатора или моста удаленные системы соединяются по телефонной линии.

Но, несмотря на то, что многие телефонные компании США уже установили коммутаторы, совместимые с ISDN, они до сих пор не решили, как продавать и обеспечивать эти услуги[10]. Позвольте дать небольшой совет: убедитесь в том, что имеете дело с отделением телефонной компании, которое предоставляет бизнес-услуги, поскольку именно к этой категории обычно относят подключение к ISDN. Не исключено, что вам придется преодолеть несколько линий клерков, прежде чем удастся добраться до кого-либо, кто хотя бы слышал об ISDN (даже если эта служба действительно предоставляется).

14.8. DSL и кабельные модемы: всенародная глобальная сеть

Для крупных компаний не составляет особого труда перемещать большие объемы данных. Такие технологии, как T1, T3, SONET, ATM и Frame Relay, реализуют достаточно простые каналы обмена данными. Но они не подходят для подключения к сети в домашних условиях. Их стоимость слишком высока, да и не всегда доступно соответствующее оборудование.

В технологии DSL (Digital Subscriber Line — цифровая абонентская линия) используется обычный медный телефонный провод, по которому передаются данные со скоростью до 7 Мбит/с (правда, для типичного DSL-соединения этот показатель находится в диапазоне от 256 кбит/с до 3 Мбит/с). Поскольку в большинстве квартир телефонная проводка уже существует, DSL — вполне оправданное последнее связующее звено между телефонной компанией и квартирой клиента. Со стороны пользователя DSL-линия заканчивается в устройстве, работающем подобно маршрутизатору TCP/IP. К нему

[10] Отсюда такой вариант расшифровки аббревиатуры ISDN: “It Still Does Nothing”, т.е. “она все еще ничего не делает”.

подключаются локальные Ethernet-устройства. Технология DSL дешевле и быстрее, чем ISDN, поэтому сегодня именно ее предпочитают домашние пользователи.

В отличие от обычных телефонных линий и ISDN-соединений, требующих "дозваниваться" до абонента, в большинстве реализаций DSL клиенту предоставляется постоянно действующая выделенная линия связи. Это делает технологию DSL еще более привлекательной, поскольку отсутствуют задержки, связанные с начальным конфигурированием и дозвоном.

Существует несколько разновидностей DSL, поэтому название технологии часто приводят в виде xDSL, где x — префикс разновидности, например: A (асимметричная), S (симметричная), H (высокоскоростная), RA (с адаптируемой скоростью) и I (DSL-через-ISDN, применяется при слишком большом удалении от центральной АТС, что не позволяет использовать более быстрые разновидности DSL). Вариант реализации и доступная скорость передачи зависят от оборудования, установленного в центральной АТС телефонной компании или провайдера Internet.

Подключение к сети сотен миллионов домашних пользователей — заветная мечта многих компаний. Здесь замешаны большие деньги. Технология DSL основана на существующей инфраструктуре телефонных линий, инвестиции в которую позволили телефонным компаниям и провайдерам-монополистам извлекать сказочные прибыли, пока мимо них проносилась сетевая революция 80-х — 90-х гг.

Компании кабельного телевидения, развивающие собственную оптоволоконную инфраструктуру, предлагают высокоскоростные (хоть и асимметричные) решения для пользователей. В индустрии кабельных модемов не так давно начался процесс стандартизации протоколов передачи данных, и сегодня рекламируется стандарт DOCSIS (Data Over Cable Service Interface Specification — спецификация интерфейса передачи данных по кабельным линиям). Он определяет технические характеристики как кабельных модемов, так и оборудования компании, предоставляющей соответствующие услуги, и позволяет взаимодействовать устройствам различных производителей.

В целом борьба между технологиями кабельных модемов и DSL сводится к тому, кто вкладывает больше средств в продвижение своей продукции на рынок. Если сравнивать сами эти технологии, то соединения являются частными для абонентов, тогда как пользователи кабельных модемов делят канал передачи со своими соседями. Поэтому DSL-линии гораздо боле безопасны.

14.9. Будущее сетей

Когда знакомишься со всеми описанными выше технологиями, замечаешь одну общую закономерность: простые и недорогие решения побеждают, а сложные и дорогие — быстро уходят в небытие. Куда же, в конце концов, приведет нас нить Ариадны?

Технология Ethernet победила своих конкурентов благодаря необычайной дешевизне. Она столь проста в реализации, что сегодня можно даже купить микроволновую печь, оснащенную интерфейсом Ethernet. Топологически эта технология хорошо масштабируется: во многих организациях старые 10-мегабитные сети, проложенные еще в начале 80-х гг., до сих пор функционируют, соединяясь с сегментами, работающими на скоростях 100 Мбит/с и 1 Гбит/с. Уже доступны оптоволоконные сети Ethernet с пропускной способностью 10Гбит/с, и вскоре они получат повсеместное распространение наряду с медными кабелями. Мы предполагаем, что эта тенденция сохранится, и все более быстродействующие коммутаторы будут по-прежнему поддерживать различные варианты Ethernet.

Что касается подключения домашних пользователей, то технология DSL вдохнула новую жизнь в старые медные рудники. Широкое распространение кабельных модемов позволило миллионам семей получить высокоскоростной доступ к сети.

Во всех этих новых разработках прекрасно то, что, несмотря на тип среды и скорость передачи данных, протоколы TCP/IP применимы везде.

14.10. Тестирование и отладка сетей

Основной причиной широкомасштабного перехода к стандарту Ethernet (и к другим технологиям, основанным на использовании неэкранированной витой пары) явилась простота отладки сети. Поскольку эти сети можно проверять посегментно, аппаратные проблемы часто решаются в считанные секунды.

Ключ к отладке сети — ее разбивка на сегменты и тестирование каждого из них до тех пор, пока не будет обнаружено неисправное устройство или кабель. Загадочные лампочки на коммутаторах и концентраторах (обозначающие, к примеру, состояние канала и наличие трафика пакетов) помогают быстро выявить источник проблемы.

Как всегда, важно иметь под рукой нужные инструменты, чтобы выполнить работу правильно и без проволочек. На рынке предлагаются средства сетевой отладки двух типов (правда, наблюдается тенденция к их объединению).

Устройство первого типа — ручной кабельный тестер. Он измеряет электрические характеристики кабеля, включая его длину (для этого применяется особая технология, называемая *рефлектометрией во временной области*). Такие устройства способны выявлять простейшие проблемы, например разрыв или неправильную разводку кабеля. Нашим любимым инструментом тестирования локальных сетей является устройство Fluke LanMeter. Это универсальный анализатор, способный даже посылать эхо-пакеты протокола ICMP. Для глобальных (телекоммуникационных) сетей наш фаворит — устройство T-Berd. Оно выпускается компанией Acterna (`www.acterna.com`), как и его более совершенный ЛВС-аналог FIREBERD.

Средства отладки второго типа — это анализаторы пакетов. Они просматривают сетевые пакеты на предмет наличия ошибок протоколов, неправильной конфигурации и прочего беспорядка. Существуют коммерческие анализаторы, но мы пришли к выводу, что в большинстве случаев наилучший вариант — бесплатная программа Wireshark (`www.wireshark.org`), которая работает даже на медленных портативных компьютерах[11].

14.11. Прокладка кабелей

При прокладке кабелей Ethernet, гигабитных или обычных, рекомендуется использовать провод самого высокого качества. Это позволит пользоваться им даже спустя десять лет. Наиболее дешевый вариант — проложить кабели сразу во всем здании, а не заниматься этим по мере необходимости.

Кабели НВП

Как правило, кабель категории 5E имеет наилучшее соотношение цена/производительность на современном рынке. Его стандартный вариант — четыре пары проводов

[11] Подобно многим популярным программам Wireshark часто подвергается атакам со стороны хакеров. Убедитесь, что вы используете наиболее свежую версию.

под одной оболочкой, что подходит для большинства соединений, включая RS-232 и гигабитные линии.

Спецификации кабеля категории 5E требуют, чтобы скрутка провода заканчивалась не более чем за полдюйма до соединения с монтажным блоком. Это означает, что кабель с более чем четырьмя парами проводов под одной оболочкой необходимо дополнительно обматывать или защищать, так как он предназначен для обслуживания нескольких соединений.

С кабелем категории 5E должны использоваться соответствующие оконечные разъемы. Мы пользуемся продукцией компании Siemon.

Офисные точки подключения

Одной точки подключения на офис явно недостаточно. Сколько же нужно: две или четыре? Мы рекомендуем четыре, обосновывая это следующими причинами.

- К этим точкам можно подключать последовательные устройства (модем, принтер и т.д.).
- Их можно использовать просто для подключения телефонов.
- Их можно применять для подключения портативных или демонстрационных компьютеров.
- Стоимость материалов составляет, как правило, всего 5–10% от общей стоимости прокладки сети.
- Зачастую достаточно вдвое увеличить предполагаемые затраты.
- Гораздо дешевле проложить весь кабель сразу, чем делать это поэтапно.
- Когда имеющиеся порты перестают справляться с нагрузкой, пользователи приобретают 10 Мбит/с концентраторы в ближайшем магазине, а затем начинают жаловаться на недостаточную скорость подключения.

При прокладке кабеля по зданию можно установить дополнительные розетки в коридорах, конференц-залах, столовых и ванных комнатах, а также потолочные беспроводные точки доступа. Однако при этом следует помнить о безопасности и подключать общедоступные порты к “гостевым” ВЛВС, не имеющим доступа к ресурсам внутренней сети.

Стандарты кабельных систем

Необходимость обеспечения всех видов деятельности внутри современных зданий обусловливает потребность в крупной и сложной кабельной инфраструктуре. Заглянув в обычный коммутационный шкаф, вы будете потрясены, увидев его стенки, сплошь покрытые непомеченными проводами одного цвета.

С целью улучшения прослеживаемости и стандартизации кабельных систем зданий в феврале 1993 г. организация TIA опубликовала административный стандарт на телекоммуникационную инфраструктуру коммерческих зданий (TIA/EIA-606). Этот стандарт устанавливает требования и принципы идентификации и документирования телекоммуникационной инфраструктуры.

Он касается следующих аспектов:

- оконечной аппаратуры;
- кабелей;

- прокладки кабелей;
- расстояний между элементами оборудования;
- цветовой маркировки;
- символических обозначений стандартных компонентов.

В частности, определены стандартные цвета маркировки проводов (табл. 14.5).

Таблица 14.5. Таблица цветовой маркировки по стандарту TIA/EIA-606

Тип оконечного устройства	Цвет	Код[а]	Комментарии
Граничное	Оранжевый	150C	Центральная телефонная станция
Сетевые соединения	Зеленый	353C	Также применяется для вспомогательных электросетей
Общее оборудование[б]	Фиолетовый	264C	Основное оборудование коммутации и передачи данных
Магистраль первого уровня	Белый	—	Кабели
Магистраль второго уровня	Серый	422C	Кабели
Станция	Синий	291C	Горизонтальные кабели
Магистраль между зданиями	Коричневый	465C	Кампусные кабели
Разное	Желтый	101C	Служебные и сигнальные линии
Ключевые телефонные системы	Красный	184C	—

[а] В соответствии с цветовой моделью Pantone.

[б] Офисные АТС, компьютеры, ЛВС, мультиплексоры и т.д.

Компания Pantone продает программное обеспечение преобразования цветов между различными цветовыми системами. Эти программы помогут идентифицировать цветовую маркировку проводов, униформы монтажников и монтажной документации.

14.12. Проектирование сетей

В этом разделе рассматриваются вопросы, связанные с логическим и физическим проектированием сетей средней величины. Представленные здесь идеи подходят для нескольких сотен узлов, но неприменимы ни для трех, ни для трех тысяч компьютеров, включенных в одну сеть. Также предполагается, что работа будет начата с нуля.

Основной объем работ по проектированию сети состоит из определения:

- типов сред передачи;
- топологии и способов прокладки кабелей;
- системы коммутаторов и маршрутизаторов.

Еще один ключевой вопрос проектирования сети связан с управлением перегрузкой. Например, файловые системы NFS и CIFS очень сильно загружают сеть, поэтому нежелательно подключать их по магистральному кабелю.

Ниже анализируются аспекты, которые необходимо учитывать при проектировании сети.

Структура сети и архитектура здания

Структуру сети проще изменить, чем архитектуру здания, но обе они должны нормально сосуществовать. Если вам крупно повезло, т.е. представилась возможность проектировать сеть до постройки здания, не скупитесь. К сожалению, в большинстве случаев здание и отдел технического обслуживания компании на момент проектирования сети уже существуют и налагают жесткие ограничения на структуру сети.

В готовых зданиях сеть должна адаптироваться к архитектуре, а не противостоять ей. В современных зданиях помимо высоковольтной электропроводки, водо- и газопроводов иногда имеются каналы для прокладки кабелей. Часто монтируются подвесные потолки — истинный подарок для тех, кто прокладывает сеть. Во многих университетских городках существуют туннели, которые облегчают создание сетей.

Необходимо следить за целостностью брандмауэров[12]. При прокладке кабеля через брандмауэр отверстие должно соответствовать диаметру кабеля и заполняться негорючим веществом. Выбирая кабель, учитывайте наличие приточной вентиляции. Если узнают, что вы нарушили правила пожарной безопасности, вас могут оштрафовать и заставить устранить недостатки, даже если для этого придется проложить заново всю сеть.

Логическая структура сети должна соответствовать физическим ограничениям зданий, в которых она будет функционировать. Приступая к проектированию, помните, что можно найти логически красивое решение, а затем вдруг обнаружить, что реализовать его физически сложно или вообще невозможно.

Существующие сети

Излагаемый здесь материал касается компьютерных сетей, но во многих организациях уже используются сети кабельного телевидения и телефонные сети, по которым можно передавать данные. Среди них часто встречаются волоконно-оптические линии связи. Если ваша организация готовится к установке новой телефонной системы, купите побольше оптоволокна и попросите проложить его одновременно с телефонным кабелем.

Нам представилась такая возможность несколько лет назад. Мы спросили подрядчиков, не могли бы они проложить для нас немного оптоволокна. Они ответили: “Конечно, и даже бесплатно”, а затем слегка опешили, когда мы появились с грузовиком, полным кабеля.

Расширение сетей

Прогнозировать потребности на десять лет вперед очень сложно, особенно в области вычислительной техники и сетей. Поэтому, проектируя сеть, всегда следует учитывать перспективы ее расширения и увеличения пропускной способности. Прокладывая кабель, особенно в труднодоступных местах, протягивайте в три-четыре раза больше пар, чем нужно. Помните: основная часть стоимости прокладки сети приходится на оплату труда, а не на материалы.

Даже если использование волоконно-оптические линии не планируется, разумно будет все же проложить немного оптоволокна, особенно если известно, что впоследствии

[12] Речь идет о брандмауэрах в их первоначальном смысле: бетонных, кирпичных или огнеупорных стенах, которые препятствуют распространению огня по всему зданию. Значительно отличаясь от сетевых брандмауэров, они не менее важны.

протянуть его будет гораздо труднее. Используйте и многомодовые, и одномодовые волоконно-оптические кабели, поскольку в будущем может потребоваться именно тот тип кабеля, который отсутствует.

Перегрузка

Сеть — как цепь: ее качество определяется самым слабым или самым медленным звеном. Производительность Ethernet, как и многих других сетевых технологий, нелинейно снижается при увеличении нагрузки.

Активно эксплуатируемые коммутаторы, нестыкующиеся интерфейсы, низкоскоростные каналы связи — все это может привести к перегрузке. Эффективный способ борьбы с ней заключается в локализации трафика путем создания подсетей и установки маршрутизаторов. Подсети можно использовать также для изоляции компьютеров, предназначенных для проведения различных экспериментов. Трудно проводить эксперимент на нескольких компьютерах, если нет надежного способа изолировать их физически и логически от остальной части сети.

Обслуживание и документирование

Опыт показывает, что удобство обслуживания сети напрямую зависит от качества документации на нее. Точная, полная, своевременно корректируемая документация абсолютно необходима.

Кабели следует маркировать во всех точках подключения нагрузки и через каждые один-два метра, чтобы их было легко узнать[13]. Рекомендуем вкладывать копии местных монтажных схем в коммутационные шкафы, чтобы при всех изменениях эти экземпляры можно было откорректировать на месте. Каждые несколько недель необходимо переносить все корректировки в базу данных прокладки кабелей.

Стыки между крупными системами в виде коммутаторов или маршрутизаторов могут упростить отладку, поскольку позволяют изолировать части сети и отлаживать их по отдельности. Полезно также разграничивать административные области.

14.13. Управление сетью

Если необходимо обеспечить нормальную работу сети, одни функции управления следует централизовать, другие — распределить, а третьи — оставить на локальном уровне. Требуется сформулировать и согласовать обоснованные “правила поведения добропорядочных граждан”.

Типичная крупномасштабная среда включает в себя:

- магистральную сеть, соединяющую здания;
- сети подразделений, подключенные к магистрали;
- подсети рабочих групп в рамках подразделения;
- соединения с внешним миром (например, с Internet или периферийными филиалами).

[13] Некоторые производители поставляют катушки с уже промаркированным через каждый метр кабелем.

При проектировании и реализации сетей следует предусматривать централизованные контроль, ответственность, сопровождение и финансирование. Поскольку подразделения, как правило, стремятся свести к минимуму собственные расходы, быстро растет число сетей с централизованной оплатой каждого соединения. Вот основные объекты централизованного управления:

- структура сети, в том числе принципы использования подсетей, маршрутизаторов, коммутаторов и т.д.;
- магистральный кабель, в том числе подключения к нему;
- IP-адреса и имена компьютеров, доменные имена;
- используемые протоколы (требуется обеспечить их взаимодействие);
- правила доступа в Internet.

Имена доменов, IP-адреса и сетевые имена компьютеров в определенном смысле уже находятся под централизованным контролем таких организаций, как ARIN и ICANN, но координация использования этих элементов на локальном уровне также необходима.

Центральный орган управления имеет общее представление о сети, ее структуре, производительности и перспективах роста. Он может позволить себе иметь собственное контрольное оборудование (и обслуживающий его персонал) и следить за нормальной работой магистральной сети. Центральный орган может настоять на правильном выборе структуры сети, даже если для этого придется заставить подразделение приобрести маршрутизатор и создать подсеть для подключения к магистрали. Такое решение иногда необходимо для того, чтобы новое соединение не навредило работе существующей сети.

Если в сети работают разнородные компьютеры, операционные системы и протоколы, обязательно нужно иметь "высокоинтеллектуальный" маршрутизатор (например, компании Cisco), который будет служить шлюзом между сетями.

14.14. Рекомендуемые поставщики

Занимаясь пятнадцать с небольшим лет инсталляцией сетей по всему миру, мы не раз обжигались на продуктах, которые не соответствовали спецификациям, имели завышенную цену, неправильно указанные характеристики или как-то иначе не оправдывали ожидания. Ниже приведен список поставщиков, которым мы доверяем и услугами которых рекомендуем пользоваться.

Кабели и разъемные соединения

AMP (now part of Tyco)
(800) 522-6752
`www.amp.com`

Anixter
(800) 264-9837
`www.anixter.com`

Belden Cable
(800) 235-3361
(765) 983-5200
`www.belden.com`

Black Box Corporation
(724) 746-5500
`www.blackbox.com`

Newark Electronics
(800) 463-9275
`www.newark.com`

Siemon
(860) 945-4395
`www.siemon.com`

Тестовые приборы

Fluke	Acterna	Siemon
(800) 443-5853	(866) 228-3762	(860) 945-4395
www.fluke.com	www.acterna.com	www.siemon.com

Маршрутизаторы/коммутаторы

Cisco Systems
(415) 326-1941
www.cisco.com

14.15. Рекомендуемая литература

- Barnett, David, David Groth, and Jim McBee. *Cabling: The Complete Guide to Network Wiring (3rd Edition)*. San Francisco: Sybex, 2004.
- Seifert, Rich. *Gigabit Ethernet: Technology and Applications for High Speed LANs*. Reading, MA: Addison-Wesley, 1998.
- ANSI/TIA/EIA-568-A, *Commercial Building Telecommunications Cabling Standard*, и ANSI/TIA/EIA-606, *Administration Standard for the Telecommunications Infrastructure of Commercial Buildings*. Это стандарты телекоммуникационной промышленности для построения кабельных систем зданий. К сожалению, они доступны не бесплатно. См. www.tiaonline.org.
- Spurgeon, Charles. *Guide to Ethernet*. www.ethermanage.com/ethernet.

14.16. Упражнения

14.1. Сегодня для прокладки компьютерных сетей в офисных зданиях чаще всего используется неэкранированная витая пара. Для поддержки таких сетей требуется определенное сочетание концентраторов и коммутаторов. Во многих случаях устройства обоих типов оказываются взаимозаменяемыми. Перечислите достоинства и недостатки этих устройств.

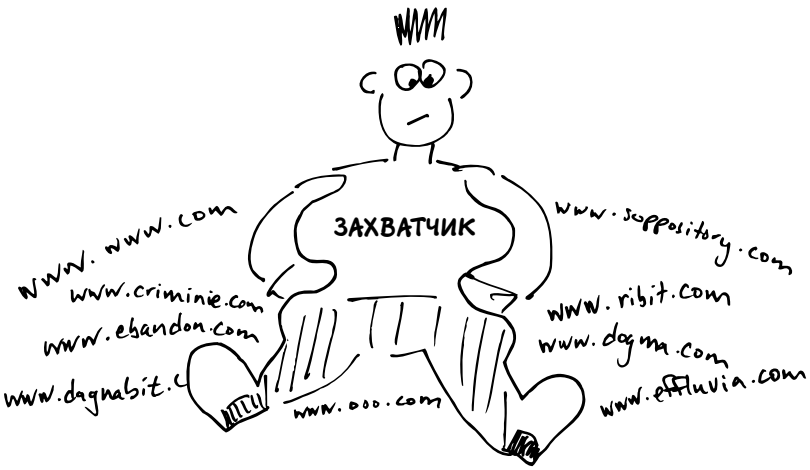
☆ 14.2. Нарисуйте схему воображаемой сети, соединяющей компьютер вашей лаборатории с узлом Amazon.com. Покажите компоненты локальных, общегородских и глобальных сетей и перечислите технологии, применяемые на каждом уровне. Укажите, где могут использоваться концентраторы, коммутаторы и маршрутизаторы.

☆ 14.3. Проанализируйте временный протокол целостности ключей WPA. Подробно опишите его преимущества по сравнению с протоколом WEP и типы атак, которые он предотвращает.

☆☆ 14.4. TTCP — средство измерения производительности протоколов TCP и UDP (www.rpmfind.net). Инсталлируйте пакет на двух компьютерах и замерьте производительность соединения между ними. Что происходит с пропускной способностью сети при изменении размеров сетевых буферов? Как полученные результаты согласуются с теоретической пропускной способностью физической среды?

Глава 15

Система доменных имен



К Internet подключено множество компьютеров. Как управлять ими, если они принадлежат к разным странам, сетям и административным группам? Этой цели служат два элемента глобальной сетевой инфраструктуры: система доменных имен (Domain Name System, DNS), которая отслеживает информацию об именах и адресах компьютеров, и система маршрутизации в Internet, контролирующая соединения между компьютерами.

Настоящая глава (ее размеры таковы, что о ней впору говорить как о небольшой книге) посвящена DNS. Эта система предназначена для нескольких целей, основная из которых — преобразование имен компьютеров в IP-адреса и наоборот. Пользователям и программам пользовательского уровня удобнее ссылаться на компьютеры по именам, но низкоуровневое сетевое программное обеспечение понимает только числовые адреса. DNS играет роль промежуточной связующей системы. Она также важна для маршрутизации электронной почты, организации доступа к Web-серверам и выполнения многих других функций.

DNS представляет собой распределенную базу данных. Термин “распределенная” означает, что сведения о компьютерах хранятся на серверах, которые автоматически вступают в контакт друг с другом, запрашивая данные и обмениваясь информацией.

Мы будем изучать DNS в три этапа.

1. Общий анализ системы DNS: ее история, данные и протоколы.
2. BIND — особая реализация системы DNS.
3. Функционирование и обслуживание серверов BIND, а также рассмотрение смежных вопросов, таких как безопасность.

Если вам нужно настроить или обслуживать серверы DNS вашей организации, и вы уже имеете общее представление о работе DNS, можете смело пропустить этот материал.

Прежде чем мы перейдем к изучению основ DNS, позвольте сначала уйти немного в сторону от этой темы и ответить на наиболее часто задаваемый вопрос: как подключить новый компьютер к сети, в которой уже используется BIND? Представленный ниже материал можно сравнить с рецептом, взятым из поваренной книги. В нем вы не найдете определения или объяснения какого-либо термина; более того, он вряд ли подойдет в точности к вашим локальным условиям. Перечисленные ниже действия нужно выполнять с особым вниманием.

15.1. DNS для нетерпеливых: подключение нового компьютера

Если ваша сеть настроена на использование протокола DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования хоста), то вам вряд ли придется что-либо конфигурировать вручную специально для DNS. При подключении нового компьютера сервер DHCP сообщит ему о том, какие серверы DNS должен будет использовать этот компьютер для запросов. Преобразования имени компьютера в IP-адрес, используемый во внешнем мире, осуществлялись главным образом при конфигурировании сервера DHCP и автоматически вводятся с помощью средства динамического обновления DNS.

Для сетей, в которых DHCP не используется, этот рецепт показывает, каким образом можно обновить конфигурацию DNS, копируя и изменяя записи в похожем компьютере.

Шаг 1. Выберите для нового компьютера сетевое имя и IP-адрес, согласовав их с системным администратором или провайдером Internet.

Шаг 2. Найдите похожий компьютер в той же подсети. Вы будете создавать новые записи на основе записей этого компьютера. В качестве модели мы будем использовать в данном примере компьютер `templatehost.my.domain`.

Шаг 3. Зарегистрируйтесь на компьютере, который является главным сервером имен.

Шаг 4. Просмотрите конфигурационный файл сервера имен. Обычно это файл `/etc/named.conf`.

- Найдите в инструкции `options` строку `directory`, где сообщается о местонахождении зонных файлов в системе (см. раздел 15.11). В этих файлах хранятся реальные имена и IP-адреса компьютеров.
- Найдите в инструкциях `zone` имена файлов зон прямого и обратного преобразований для сети, к которой принадлежит новый IP-адрес (см. раздел 15.11).
- Убедитесь в том, что в инструкциях `zone` данный сервер указан как главный сервер домена. Инструкция зоны прямого преобразования должна выглядеть примерно так:

```
zone "my.domain"{
    type master;
...
```

Шаг 5. Перейдите в каталог файлов зон и отредактируйте файл зоны прямого преобразования. Найдите записи, соответствующие компьютеру-прототипу. Они будут выглядеть примерно так:

```
templatehost    IN   A    128.138.243.100
                IN   MX   10 mail-hub
                IN   MX   20 templatehost
```

В вашем варианте может не быть строк с обозначением `MX`, которые применяются для маршрутизации почтовых сообщений.

Шаг 6. Скопируйте эти записи и модифицируйте их должным образом для нового компьютера. Записи зонного файла могут быть отсортированы по именам компьютеров, поэтому следуйте принятым соглашениям. Не забудьте изменить также порядковый номер записи SOA, расположенной в начале файла (это первый из пяти числовых параметров записи). Порядковый номер может только возрастать. Прибавьте к нему единицу, если порядковые номера выбираются произвольно, или запишите в это поле текущую дату, если в системе используется такое соглашение.

Шаг 7. Откройте файл зоны обратного преобразования[1], скопируйте запись, соответствующую компьютеру-прототипу, и отредактируйте ее. Она будет выглядеть примерно так:

```
100             IN   PTR  templatehost.my.domain.
```

Обратите внимание на то, что после имени компьютера стоит точка; ее нельзя пропускать. Необходимо также обновить порядковый номер записи SOA в данном файле.

Если в файле зоны обратного преобразования приводится не только последний байт IP-адреса хоста, то числа нужно вводить в обратном порядке. Например, запись

```
100.243         IN   PTR  templatehost.my.domain.
```

соответствует IP-адресу 128.138.243.100 (здесь зона обратного преобразования относится к домену 138.128.in-addr.arpa, а не 243.138.128.in-addr.arpa).

Шаг 8. Не выходя из системы, выполните команду `rndc reload`, или, если сервер занят, просто перезагрузите домены (или виды), которые были изменены:

```
# rndc reload имя_зоны_прямого_преобразования
# rndc reload имя_зоны_обратного_преобразования
```

Шаг 9. Проверьте конфигурацию DNS с помощью команды `dig` (описана в разделе 15.16). Попробуйте также выполнить команду `ping` или `traceroute`, задав имя нового компьютера, даже если он еще не сконфигурирован. Сообщение “host unknown” (неизвестный хост) свидетельствует о наличии ошибки. Сообщение “host not responding” (хост не отвечает) означает, что все в порядке.
Чаще всего совершают такие ошибки:

- забывают обновить порядковый номер и перезапустить сервер имен;
- забывают добавить точку в конец доменного имени в записи PTR.

[1] Этот файл может находиться где угодно, например у провайдера. В этом случае изменения придется вносить там.

15.2. История DNS

Систему доменных имен формально описал Пол Мокапетрис (Paul Mockapetris) в документах RFC882 и RFC883 (1983 г.), обновив спецификацию в документах RFC1034 и RFC1035 (1987 г.). Эта система была построена на двух ключевых концепциях: иерархических именах хостов и распределенной ответственности.

Реализации BIND

Работа, связанная с реализацией DNS в UNIX, была начата в 1984 г. четырьмя аспирантами университета Беркли: Дугласом Терри (Douglas Terry), Марком Пейнтером (Mark Painter), Дэвидом Ригглом (David Riggle) и Сонг Ньян Чжоу (Songnian Zhou). В середине 1980-х этот проект был включен в дистрибутив Berkley UNIX Кевином Данлапом (Kevin Dunlap) и стал известен как BIND (Berkeley Internet Name Domain — система доменных Internet-имен реализации университета Беркли). В настоящее время за сопровождение BIND отвечают Пол Викси (Paul Vixie) и ISC (Internet Systems Consortium — консорциум Internet-систем, до 2004 г. известный как Internet Software Consortium — консорциум разработчиков программного обеспечения для Internet; Web-страница консорциума находится по адресу `www.isc.org`). Этот проект имеет открытый исходный код. В 2000 и 2001 гг. в ISC была разработана совершенно новая версия BIND: BIND 9. Она была профинансирована различными фирмами-производителями, правительственными организациями и прочими субъектами.

Организация ISC также предоставляет пользователям своих продуктов различные виды технической поддержки, включая помощь в настройке и даже программировании. Проводятся курсы по обучению работе с BIND и DNS. Это очень удобно для тех организаций, которые соглашаются использовать ПО с открытым исходным кодом только при условии предварительного заключения контракта на техническую поддержку. Некоторые компании оказывают консорциуму финансовую помощь посредством заключения дорогостоящих контрактов на обслуживание, хотя реально они в нем не нуждаются.

Компания Nortel перенесла пакет в среду Windows, поэтому теперь BIND доступна для платформ Windows, UNIX и Linux. Благодаря стандартизации протокола DNS различные серверы DNS, работающие как в UNIX, так и в других операционных системах, могут свободно взаимодействовать и обмениваться данными друг с другом. Во многих организациях имеются UNIX-серверы, предоставляющие услуги DNS рабочим станциям Windows.

Документы RFC1034 и RFC1035 до сих пор считаются базовыми спецификациями DNS, хотя за последнее десятилетие появилось более 40 других документов (см. список в конце главы), видоизменивших или дополнивших DNS. В настоящее время не существует единого стандарта, в котором все эти компоненты были бы сведены воедино. Традиционно определение DNS звучало примерно так: “Это то, что реализует пакет BIND”. Но такое определение становится все менее точным по мере того, как появляются другие серверы DNS.

Другие реализации DNS

С самого начала система BIND являлась всего лишь реализацией DNS. В наши дни существует несколько версий системы, как коммерческих, так и бесплатных (с открытым кодом). Во многих из них нет всех тех спецификаций, которые определены в доку-

ментах RFC о DNS и которые, по сути, являются стандартами. В табл. 15.1 перечислены наиболее популярные реализации BIND и указаны адреса Web-страниц, на которых можно найти подробные сведения.

Таблица 15.1. Некоторые популярные реализации DNS

Название	Автор	Источник	Комментарии
BIND	ISC	`isc.org`	Авторитетная или кэширующая
NSD	NLnet Labs	`www.nlnetlabs.nl`	Только авторитетная.
PowerDNS	PowerDNS BV	`www.powerdns.com`	Только авторитетная.
djbdns[a]	Dan Bernstein	`tinydns.org`	Не согласована с некоторыми документами RFC.
Microsoft DNS	Microsoft	`microsoft.com`	Виновата в неисчислимых грехах.
ANS, CNS	Nominum	`www.nominum.com`	Авторитетная или кэширующая

[a] Эта версия известна также как tinydns, которая является серверным компонентом пакета djbdns.

ISC постоянно следит за различными реализациями DNS и определенным количеством серверов имен. Чтобы получить сведения о "демографической обстановке", перейдите на страницу `www.isc.org`, щелкните на ссылке ISC Internet Domain Survey, затем щелкните на ссылке Latest Survey Results, а затем — на ссылке Domain Server Software Distribution. Такие варианты применения DNS, как Infoblox (`www.infoblox.com`), используются в нескольких крупных организациях, хотя в обзоре их пока что нет.

В этой книге мы будем говорить лишь о BIND, которая считается эталонной реализацией DNS. В настоящее время она приобрела широкое применение и идеально подходит для многих сайтов. BIND придерживается всех существующих стандартов, в том числе и стандартов, предложенных комитетом IETF (Internet Engineering Task Force — комитет, занимающийся развитием архитектуры Internet), и часто реализует такие возможности, для которых еще нет полной спецификации. Это очень неплохо, поскольку так можно тестировать новые особенности, выявляя и устраняя в них проблемы до того, как особенности будут приняты в качестве стандарта.

NSD — демон сервера имен (name server daemon) — был разработан в 2003 г. фирмой NLnet Labs, базирующейся в Амстердаме. Это быстрый, безопасный и авторитетный сервер имен, подходящий для использования как в серверах корневых доменов, так и в серверах доменов верхнего уровня. (Авторитетный сервер подходит для ответа на вопросы о компьютерах в вашем домене, но он не может отвечать на вопросы пользователей о других доменах.)

PowerDNS — авторитетный сервер имен с открытым кодом, обеспечивающий уникальную гибкую серверную систему. Данные DNS можно брать из файлов или других ресурсов, таких как MySQL, Oracle (8i и 9i), DB2 фирмы IBM, PostgreSQL, SQL Server фирмы Microsoft, LDAP, ODBC, XDB и даже из канала UNIX.

djbdns — альтернативный пакет сервера имен, состоящий из авторитетного сервера tinydns и кэширующего сервера dnscache. Судя по отзывам, он функционирует безопасно и очень быстро, хотя у нас есть сведения о том, что его разработка до конца еще не завершена. Основной его недостаток заключается в том, что он часто и намеренно нарушает стандарты DNS, что приводит к осложнению взаимодействия с другими серверами DNS.

Microsoft предлагает сервер DNS для Windows, хотя в нем тоже есть свои сложности и отличия. Этот сервер может взаимодействовать с BIND, однако он часто "засоряет" сеть ненужными и деформированными пакетами.

Фирма Nominum, создавшая первоначальную версию BIND 9 для ISC, продает свои собственные серверы имен и средства управления сетью. Серверы Nominum характеризуются высокой скоростью работы и соответствуют большинству существующих на данный момент стандартов.

15.3. Основные задачи DNS

DNS определяет:

- иерархическое пространство имен компьютеров и IP-адресов;
- таблицу имен компьютеров и адресов, реализованную в виде распределенной базы данных;
- "распознаватель" — клиентскую библиотеку функций, осуществляющих запросы к базе данных DNS;
- усовершенствованные средства маршрутизации электронной почты;
- механизм поиска служб в сети;
- протокол обмена информацией об именах.

Хосту, подключенному к Internet, система доменных имен нужна для полноценного участия в работе сети. Вести локальный файл `/etc/hosts` с данными о каждом компьютере, с которым когда-нибудь понадобится установить контакт, невозможно в принципе.

В каждой организации хранится один или несколько фрагментов распределенной базы данных, образующей всемирную систему доменных имен. В состав фрагмента входят текстовые файлы, содержащие записи об хостах организации. Запись — это отдельная строка, состоящая из имени компьютера, обозначения типа записи и нескольких информационных полей. Поле имени можно опустить, если оно совпадает с аналогичным значением предыдущей записи.

Например, строки

```
bark        IN   A    206.168.198.209
            IN   MX   10 mailserver.atrust.com.
```

в файле зоны прямого преобразования и строка

```
209         IN PTR   bark.atrust.com.
```

в файле зоны обратного преобразования устанавливают соответствие между хостом `bark.atrust.com` и IP-адресом 206.168.198.209. Почта, адресованная компьютеру, пересылается на хост `mailserver.atrust.com`.

DNS — это клиент-серверная система. Серверы (называемые серверами имен) загружают данные из клиентских DNS-файлов в память и пользуются ими при ответах на запросы как от внутренних клиентов, так и от внешних компьютеров. Все компьютеры, подключенные к сети, должны быть клиентами DNS, но лишь некоторым из них требуется быть серверами имен.

В небольшой организации (несколько хостов в одной сети) можно запустить сервер на одном из компьютеров или попросить Internet-провайдера предоставить услуги DNS.

Средних размеров организация с несколькими подсетями должна иметь несколько серверов DNS, чтобы сократить время выполнения запросов и повысить общую надежность сети. Очень большая организация может разделить свой домен на поддомены и запустить в каждом из них несколько серверов.

15.4. Пространство имен DNS

Пространство имен имеет такую структуру, которую в математике называют деревом; каждое доменное имя соответствует хосту (компьютеру) в дереве. Одна ветка дерева имен DNS преобразовывает имена компьютеров в IP-адреса, а другая ветка преобразовывает IP-адреса в имена компьютеров. Первая ветка называется "прямым преобразованием", а связанные с ней файлы данных BIND называются "файлами зон прямого преобразования". Вторая ветка называется "обратным преобразованием", а связанные с ней файлы данных называются "файлами зон обратного преобразования". К сожалению, во многих сайтах не поддерживается возможность обратного преобразования.

Каждый домен — это отдельный фрагмент пространства имен, находящийся под единым административным контролем. Корень дерева носит имя "." (точка); под ним находятся домены верхнего, или корневого, уровня.

По историческим причинам существуют два вида имен доменов верхнего уровня. В США такие домены изначально отражали организационно-политическую структуру глобальной сети и, как правило, носили трехбуквенные имена, например "com" или "edu". Некоторые из этих доменов (особенно "com", "org" и "net") распространены также за пределами США. Все они называются *общими* или *организационными* доменами верхнего уровня (generic Top-Level Domain, gTLD).

Домены верхнего уровня раньше были относительно постоянными, но в 2001 г. ICANN ввела семь новых доменов: "biz", "info", "name", "pro", "museum", "aero" и "coop"[2]. Позже в список gTLD был добавлен домен "jobs". Сейчас эти домены функционируют и являются доступными для использования. Домены "biz", "info" и "name" называются "нефинансируемыми" gTLD и доступны каждому. Домены "museum", "aero", "jobs", "pro" и "coop" являются финансируемыми TLD и доступны ограниченному кругу субъектов, подавших заявку на регистрацию.

В табл. 15.2 перечислены наиболее важные организационные домены и указано их первоначальное предназначение. После того как в домене "com" короткие, коммерчески привлекательные имена оказались занятыми, организации, управляющие адресными реестрами, стали продавать имена в доменах "org" и "net", не обращая внимания на существовавшие ограничения. Домены, приведенные в левой половине таблицы, являются исходными, дата их появления — 1988 год. В правой половине указаны новые домены, появившиеся в 2001 г.

Для большинства доменов вне США используются двухбуквенные коды стран, утвержденные организацией ISO. Эти домены называются *географическими* (country code Top-Level Domain, ccTLD). Домены обоих типов сосуществуют в одном глобальном пространстве имен. Некоторые коды стран перечислены в табл. 15.3.

[2] ICANN (Internet Corporation for Assigned Names and Numbers — организация по назначению имен и адресов в Internet) является руководящим органом сети Internet (см. раздел 11.8).

Таблица 15.2. Организационные домены верхнего уровня

Домен	Для кого предназначен	Домен	Для кого предназначен
com	Коммерческие компании	aero	Авиационная промышленность
edu	Учебные заведения	biz	Торговые предприятия
gov	Правительственные учреждения	coop	Совместные предприятия
mil	Военные ведомства	info	Неограниченное использование
net	Поставщики сетевых услуг	jobs	Трудоустройство
org	Некоммерческие организации	museum	Музеи
int	Международные организации	name	Физические лица
arpa	Корень дерева IP-адресов	pro	Бухгалтерские, адвокатские фирмы и т.п.

Таблица 15.3. Коды стран

Код	Страна	Код	Страна	Код	Страна
au	Австралия	fi	Финляндия	ch	Швейцария
ca	Канада	fr	Франция	hu	Венгрия
br	Бразилия	jp	Япония	ua	Украина
de	Германия	se	Швеция	ru	Россия

В ряде стран организационная структура глобальной сети формируется на базе доменов второго уровня, при этом правила именования могут быть разными. Например, научные учреждения в США относятся к домену “edu”, а в Японии — к домену `ac.jp`.

В США домен верхнего уровня “us” используется нечасто, в основном в сочетании с региональными доменами. Например, `bvsd.k12.co.us` — это домен школьного округа города Боулдер, штат Колорадо, где находится наш университет. Домен “us” никогда не объединяется с организационными доменами, т.е. домена `edu.us` нет (пока). Преимущество домена “us” заключается в том, что регистрация имен в нем бесплатна или стоит очень недорого (за подробностями обратитесь на Web-сайт `www.nic.us`). В настоящее время рассматривается вопрос об ослаблении ограничений на формирование доменов второго уровня в домене “us” (ранее допускались только региональные домены штатов), поэтому вскоре станут возможными доменные имена вида `evi-nemeth.us`.

Торговцы доменами иногда выкупают пространства имен целых стран. Например, домен республики Молдова, “md”, в настоящее время продается медицинским заведениям и резидентам штата Мэриленд (MD), США. Другой пример — островное государство Тувалу, код которого — “tv”. Первая сделка такого рода коснулась королевства Тонга (домен “to”). В настоящее время активнее всего распродается домен острова-колонии Ниуэ (“nu”), а одним из самых привлекательных является домен Туркменистана (“tm”). Иногда подобные сделки были выгодны этим странам, иногда — нет.

Широко практикуется также “упреждающий” захват неиспользуемых доменных имен. Предприимчивые дельцы регистрируют имена, которые, по их мнению, обязательно будут востребованы в будущем. Через некоторое время какая-нибудь ничего не подозревающая компания, попытавшись зарегистрировать одно из таких имен, с удивлением обнаружит, что оно занято. Ей придется выкупать имя у предусмотрительного

владельца. Много лет назад один бизнесмен зарегистрировал доменные имена всех лыжных курортов Колорадо и заработал большие деньги, продавая имена по мере того, как администрации лыжных курортов осваивали глобальную сеть.

Текущая стоимость хорошего имени в домене "com" колеблется от нескольких тысяч до нескольких миллионов долларов. Нам предложили $50 000 за домен `admin.com`, который мы зарегистрировали несколько лет назад после того, как оказалось, что домен `sysadmin.com` уже занят журналом "/Sys/Admin". Самая высокая цена (во всяком случае, самая высокая цена, заявленная в публичном документе) была уплачена в период ажиотажа за домен `business.com` — он был продан за 7,5 млн. долларов! Сегодня уже никого не удивляют суммы в пределах от 20 000 до 100 000 долларов, хотя нередко осуществляются сделки по продаже доменов, где счет идет на миллионы долларов; пример — продажа домена `CreditCard.com` за 2,75 миллиона долларов.

Internet-делец Ден Париси (Dan Parisi) планировал получить несколько миллионов долларов за первый сайт порнографического содержания `whitehouse.com`, зарегистрированный в начале 2004 г. С того момента под этим именем открывались сайты нескольких разных предприятий, хотя точные сведения о сроках аренды и финансовая сторона сделки не известна широкой публике.

В именах доменов допускается использование только букв, цифр и символа "тире". Каждый компонент имени может иметь до 63 символов, а само имя может иметь не более 256 символов. Локализация (привязка к стандартам страны, в которой находится домен) системы DNS и поддержка наборов символов, отличных от ASCII, когда-нибудь приведет к тому, что будут изменены все правила именования, а пока что имена, составленные из других наборов символов, преобразовываются в ASCII-имена; см. раздел 15.6.

В именах доменов не учитывается регистр. Для DNS домен "Colorado" идентичен доменам "colorado" и "COLORADO". В текущих реализациях протокол DNS должен игнорировать регистр при сравнении имен, но распространять имена в том регистре, в котором они указаны. Раньше было принято обозначать домены верхнего уровня прописными буквами и начинать с прописной буквы домены второго уровня. Сегодня нормой стали строчные буквы[3].

Полностью определенное имя компьютера, подключенного к Internet, формируется путем прибавления имени домена к локальному сетевому имени. Например, `boulder.colorado.edu` — полное имя хоста `boulder` в университете штата Колорадо. Другие организации могут свободно использовать имя `boulder`, так как полные имена их компьютеров все равно будут другими.

В системе DNS полностью определенные имена заканчиваются точкой, например "`boulder.colorado.edu.`". Отсутствие завершающей точки может свидетельствовать о неполном имени. В зависимости от контекста использования неполного имени к нему могут присоединяться различные дополнительные компоненты. Пользователи DNS обычно не подозревают о существовании хвостовой точки. Более того, некоторые системы (например, электронная почта) неправильно интерпретируют адрес, если к нему добавлена точка.

Компьютеры часто имеют несколько имен. К примеру, хосту `boulder.colorado.edu` можно также дать имя `www.colorado.edu` или `ftp.colorado.edu`, если в названии требуется отразить тип службы. Вообще это достаточно распространенная практика — де-

[3] Пакет BIND (имеется в виду его эталонная реализация, используемая в большинстве систем) сохраняет регистр символов, но есть реализации DNS (например, компании Microsoft или пакет djbdns), которые меняют регистр по своему усмотрению.

лать “служебные” имена (в частности, те, что начинаются с “www”) мобильными, чтобы серверы можно было переносить с одного компьютера на другой, не меняя основное имя компьютера. Присвоение дополнительных имен осуществляется с помощью записи CNAME (описана в разделе 15.11).

Когда нам выделили домен colorado.edu, мы получили гарантию того, что имя “colorado” уникально в пределах домена “edu”. Затем мы разделили этот домен на поддомены в соответствии со списком факультетов нашего университета. Например, компьютер `anchor` на факультете вычислительной техники носит Internet-имя anchor.cs.colorado.edu.

Создание каждого нового поддомена необходимо согласовывать с администрацией домена, чтобы гарантировать уникальность имени. Записи в конфигурационных файлах родительского домена делегируют поддомену полномочия по управлению соответствующей областью пространства имен.

Владельцы доменов

Управление доменами верхнего уровня “com”, “org”, “net” и “edu” раньше координировалось компанией Network Solutions, Inc., которая заключила контракт с Национальным научным фондом США. Теперь этой монополии пришел конец, и другие организации получили разрешение регистрировать имена в общих доменах. Что касается географических доменов, то ими управляют различные региональные организации.

Делались различные предложения по поводу того, чтобы разрешить частным компаниям управлять своими доменами верхнего уровня, поэтому вполне вероятно, что в ближайшем будущем такие домены появятся. Самую последнюю информацию можно получить на Web-сайте `www.icann.org`.

Большинство Internet-провайдеров предоставляет платные услуги по регистрации доменных имен. В ответ на поступивший запрос они связываются с администрацией домена верхнего уровня и настраивают свои DNS-серверы на поиск имен в пределах нового домена. Недостатком этого является то, что вы лишаетесь возможности напрямую управлять вашим доменом.

Чтобы управлять вашими собственными службами DNS, вы в любом случае должны координировать свои действия с провайдером. Многие Internet-провайдеры обеспечивают обратные преобразования IP-адресов внутри своих блоков CIDR. Если вы собираетесь управлять своими адресами, ваш провайдер должен перестать обслуживать эти адреса и предоставить вам возможность управлять ними.

🕮 О протоколе CIDR рассказывалось в разделе 12.4.

Прямые и обратные преобразования по возможности должны осуществляться в одном месте. Некоторые провайдеры с удовольствием отдают контроль над прямым преобразованием, но отказываются делать то же самое в отношении обратного преобразования. Подобное разделение обязанностей ведет к проблемам синхронизации. В разделе 15.11 описан изящный прием, позволяющий посредством записей CNAME управлять делегированием даже крошечных фрагментов адресного пространства.

Домены DNS должны (более того, обязаны — см. RFC1219) обслуживаться как минимум двумя серверами. Чаще всего в организации функционирует главный сервер, а серверы провайдера являются резервными. После того как система сконфигурирована, серверы провайдера автоматически загружают информацию из главного сервера организации. Изменения в конфигурации DNS отражаются на резервных серверах без вмешательства администраторов любой из сторон.

Не размещайте все серверы DNS в одной сети. Если по какой-то причине она окажется недоступной, пользователи других сетей не смогут работать. Распределите серверы DNS так, чтобы функционирование системы не зависело от единственного звена. При правильном конфигурировании DNS становится высоконадежной системой.

Выбор имени домена

Раньше мы советовали выбирать доменные имена так, чтобы они были короткими, удобными для ввода и чтобы они идентифицировали использующую их организацию. Сегодняшняя действительность такова, что все хорошие, короткие имена заняты, по крайней мере, в домене "com". Лишь часть из них была предусмотрительно скуплена, в основном же они используются реальными лицами. В 2004 г. использовалось более 60% зарегистрированных имен; исторически сложилось так, что в действительности используется менее половины зарегистрированных имен.

Лавинообразный рост числа доменов

Протокол DNS предназначался для того, чтобы можно было связать доменное имя организации с сервером имен, закрепленным за этой организацией. Эта схема должна была применяться во всех организациях, подключенных к Internet. Но сегодня сеть Internet стала элементом массовой культуры, и доменные имена есть у многих коммерческих продуктов, кинофильмов, спортивных мероприятий и т.д. Имена наподобие `twinkies.com` не связаны (напрямую) с компаниями, создавшими продукт; они используются в рекламных целях. Неясно, сможет ли DNS справиться с подобным разрастанием глобальной сети. Проблема заключается в том, что дерево имен DNS является эффективной информационной структурой лишь в том случае, когда в нем прослеживается четкая иерархия. Но если каждая организация дает сотням или тысячам своих продуктов доменные имена, расположенные у самой вершины дерева, о какой бы то ни было иерархии говорить не приходится.

На самом деле нам нужна служба каталогов, которая связывала бы названия различных торговых марок с соответствующими компаниями, предоставив DNS возможность заниматься только IP-адресами. Другое решение заключается в установлении принудительной иерархии имен (по типу `twinkies.hostess-foods.com`), однако это никогда не произойдет: все зашло слишком далеко, и компании, присутствующие в домене ".com", не собираются отказываться от своих доменных имен.

С точки зрения DNS примером для подражания является компания Sony: все ее продукты снабжаются именами в домене `sony.com`. Возможно, поиск информации о продуктах занимает чуть больше времени, но зато соблюдается строгий иерархический подход.

Регистрация домена второго уровня

Для того чтобы зарегистрировать домен второго уровня, необходимо подать заявку в администрацию соответствующего домена верхнего уровня. ICANN занимается аккредитацией многочисленных агентств в рамках проекта по совместной регистрации имен в организационных доменах. На момент написания книги существовало около 500 регистрационных бюро. Полный их список имеется на Web-сайте `www.icann.org`.

Чтобы зарегистрировать географическое имя ccTLD в Европе, нужно подать заявку в организацию CENTR (Council of European National Top-level domain Registries — Совет по реестрам европейских национальных доменов верхнего уровня). Ее Web-адрес —

www.centr.org. Там же можно узнать адреса региональных регистрационных бюро. В азиатско-тихоокеанском регионе ответственным органом является APNIC (Asia-Pacific Network Information Center — Азиатско-тихоокеанский центр сетевой информации) с Web-адресом www.apnic.net.

Для того чтобы заполнить бланки регистрации, нужно назначить ответственного технического специалиста и ответственного администратора, а также выбрать хотя бы два компьютера, которые будут серверами домена.

Создание собственных поддоменов

Процедура создания поддомена аналогична той, что используется при регистрации домена второго уровня, только центральная администрация теперь находится в пределах самой организации. Этот процесс предусматривает следующие этапы:

- выбор имени, уникального в пределах организации;
- назначение двух или более компьютеров серверами нового домена;
- согласование действий с администратором родительского домена.

Прежде чем передавать полномочия, администратор родительского домена должен убедиться в том, что серверы имен дочернего домена сконфигурированы и работают правильно. В противном случае произойдет то, что называется *напрасным делегированием* (подробнее об этом рассказывается в разделе 15.16).

15.5. Принципы работы DNS

Каждый компьютер, пользующийся услугами DNS, является либо клиентом этой системы, либо одновременно и клиентом, и сервером. Читатели, которые не планируют запускать серверы DNS, следующие несколько разделов могут пропустить, перейдя сразу к разделу 15.10. Отметим, однако, что приведенная информация позволит глубже понять принципы работы системы доменных имен.

Делегирование

Все серверы имен "знают" о существовании корневых серверов из локального конфигурационного файла. В свою очередь, корневые серверы "знают" о доменах "com", "net", "fi", "de" и других доменах верхнего уровня. Эта цепочка продолжается дальше: сервер домена "edu" "знает" о домене colorado.edu, сервер домена "com" "знает" о домене admin.com и т.д. Каждая зона может делегировать полномочия по управлению своими поддоменами другим серверам.

Рассмотрим реальный пример. Предположим, требуется узнать адрес хоста vangogh.cs.berkeley.edu, находясь на хосте lair.cs.colorado.edu. Компьютер lair просит локальный сервер имен, ns.cs.colorado.edu, найти ответ на этот вопрос. Последующие события представлены на рис. 15.1.

Числа возле стрелок определяют порядок событий, а буквы — тип транзакции (запрос, ответ или отсылка). Предполагается, что никакие из требуемых данных предварительно не кэшировались, за исключением имен и IP-адресов серверов корневого домена.

Локальный сервер имен не "знает" нужного адреса. Более того, ему ничего не известно о доменах cs.berkeley.edu, berkeley.edu и даже "edu". Он "знает" лишь некоторые серверы корневого домена и, будучи рекурсивным, запрашивает корневой домен об хосте vangogh.cs.berkeley.edu и получает отсылку на серверы в домене edu.

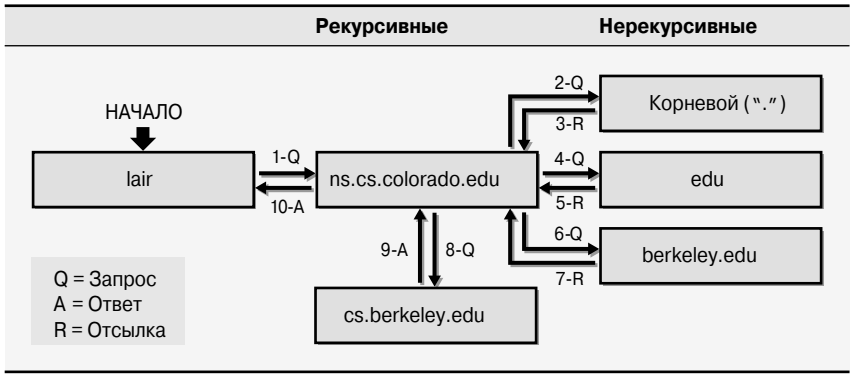


Рис. 15.1. Обработка запроса в DNS

Далее локальный сервер имен направляет запрос к серверу домена "edu" и получает ссылку на серверы домена `berkeley.edu`. Тогда он повторяет запрос, направляя его в домен `berkeley.edu`. Если сервер университета Беркли не содержит ответ в кэше, он вернет ссылку на домен `cs.berkeley.edu`. Сервер этого домена авторитетен в отношении запрашиваемой информации и возвращает адрес компьютера `vangogh`.

По окончании процедуры в кэше сервера `ns.cs.colorado.edu` окажется адрес компьютера `vangogh`. В кэше будут также находиться списки серверов доменов "edu", `berkeley.edu` и `cs.berkeley.edu`.

Кэширование и эффективность

Благодаря кэшированию повышается эффективность поиска: кэшированный ответ выдается почти мгновенно и обычно точен, так как адресно-именные соответствия меняются редко. Ответ хранится в течение определенного периода времени, называемого "временем жизни" (или "временем существования" — time to live, TTL). Большинство запросов касается локальных компьютеров и обслуживается быстро. Повышению эффективности невольно содействуют и сами пользователи, так как многие запросы повторяются.

В течение долгого времени кэширование применялось только к положительным ответам на запросы. Если имя или адрес компьютера было невозможно найти, этот факт не фиксировался. Схема кэширования отрицательных DNS-ответов была описана в документе RFC1034, но она оказалась неполной и не получила широкой реализации. Более удачный вариант схемы был описан в 1998 г. в документе RFC2308. Эта схема была реализована в BIND 8.2 в виде опционального компонента, а в BIND 9 стала неотъемлемым компонентом.

Измерения, проведенные на корневом сервере RIPE в Европе, показали, что 60% DNS-запросов относилось к несуществующим данным (многие запросы были адресованы домену 127.in-addr.arpa или содержали в качестве сетевых имен названия служб Microsoft). Кэшируя эту информацию как можно глубже в иерархии DNS, можно добиться резкого снижения нагрузки на корневые серверы.

При отрицательном кэшировании сохраняются ответы следующих типов:

- отсутствие хоста или домена, соответствующего запрашиваемому имени;
- данные запрашиваемого типа не существуют для хоста;

- запрашиваемый сервер не отвечает;
- сервер недоступен из-за проблем в сети.

Первые два типа отрицательных данных по умолчанию сохраняются в кэше в течение 10 минут. Этот период можно увеличить до трех часов с помощью параметра в записи SOA, рассматриваемой в разделе 15.7, и до одной недели с помощью опции BIND.

Во многих реализациях нет двух последних типов кэширования отрицательных ответов. Тем не менее, BIND "штрафует" серверы, не дающие ответа, и не посылает им запросы до появления других вариантов. Если ни один из серверов зоны не дает ответа, BIND ничего не помещает в кэш.

Неавторитетные запросы *могут* помещаться в кэш; авторитетные запросы *обязаны* быть помещены в кэш. Эти правила соблюдаются в BIND благодаря документам RFC, хотя в компьютерах, работающих под управлением Windows, TTL реализуется избирательным образом, по крайней мере, для отрицательного кэширования. Сначала, когда при запросе возвращается сообщение NXDOMAIN (данное доменное имя отсутствует в базе регистратора), в них используется корректное значение по умолчанию (минимальное из записи SOA), а затем TTL сбрасывается до 15 минут.

Сервер имен часто получает в ответ на запрос множество DNS-записей. Например, при попытке узнать адрес сервера имен корневого домена будет получен список из 13 корневых серверов. К какому из них следует обращаться?

Выбирая удаленный сервер среди нескольких серверов, все из которых авторитетны для домена, сервер имен BIND в первую очередь определяет полное время опроса (Round Trip Time, RTT) каждого сервера. Затем серверы разбиваются на группы в соответствии со значением RTT, после чего выбирается сервер из самой быстрой группы. Серверы в пределах одной группы считаются равнозначными и выбираются по циклической схеме.

Простейший, но весьма эффективный способ распределения нагрузки заключается в закреплении одного имени за несколькими IP-адресами (соответствующими разным компьютерам):

```
www         IN   A    192.168.0.1
            IN   A    192.168.0.2
            IN   A    192.168.0.3
```

Интенсивно эксплуатируемые Web-серверы, такие как Yahoo! или Google, в действительности не являются одним компьютером. Они просто известны под одним доменным именем в DNS[4]. Обнаружив несколько записей с одинаковым именем и типом, сервер имен возвращает всех их клиенту, но в циклическом порядке. Например, порядок записей A в показанном выше примере будет 1–2–3 для первого запроса, 2–3–1 — для второго и 3–1–2 — для третьего.

Расширенный протокол DNS

Исходная спецификация протокола DNS датируется концом 80-х гг. и предполагает использование протоколов UDP и TCP (порт 53). Первый нужен для запросов и ответов, а второй — для зонных пересылок между главными и подчиненными серверами. К сожалению, максимальный размер пакета, который гарантированно поддерживается всеми

[4] На момент написания этой книги сервер Google обслуживали 400 000 компьютеров, работающих под управлением Linux (это наша оценка), а сервер Yahoo! — более 100 000 компьютеров, работающих под управлением FreeBSD.

реализациями UDP, составляет 512 байт. Это слишком мало для новых схем обеспечения безопасности DNS (например, DNSSEC), подразумевающих включение в каждый пакет цифровой подписи.

Это ограничение влияет на число и названия корневых серверов. Чтобы все данные этих серверов умещались в 512-байтовом UDP-пакете, количество серверов ограничено числом 13, а имя каждого сервера состоит из одной буквы.

Многие распознаватели выдают сначала UDP-запрос, и если приходит усеченный ответ, то запрос повторяется в формате TCP. Такая схема позволяет обойти 512-байтовый лимит, но она неэффективна. Кое-кто может посчитать, что следует вообще отказаться от протокола UDP и перейти исключительно на TCP, но TCP-соединения намного дороже. Обмен данными между серверами имен в рамках протокола UDP может свестись всего к двум пакетам: один запрос и один ответ. В случае протокола TCP требуется минимум семь пакетов: трехфазное квитирование для инициализации диалога, запрос, ответ и завершающее квитирование для разрыва соединения.

15.6. Что нового в DNS

Последние разработки в области DNS касаются больше вопросов политики, чем техники. VeriSign — компания-регистратор, монополист в области регистрации доменных имен, которая в настоящее время отвечает за зоны “com” и “net”, добавила для этих зон адресную метазапись (wild card address record). После этого каждый пользователь, допустивший ошибку при вводе доменного имени, попадал на сайт, поддерживаемый одним из рекламодателей VeriSign. Эта служба известна под именем Site Finder (Искатель сайтов).

Пользователям Internet это пришлось не по духу, поэтому ISC включила в BIND опцию `delegation-only`, которая распознает подобные ситуации и возвращает ответ “no such domain” (такой домен отсутствует), а не направляет по адресам, которые устанавливает VeriSign. Это исправление подошло для большинства доменов верхнего уровня, но не для всех, поэтому для более точного управления впоследствии были добавлены новые опции BIND, рассматриваемые в разделе 15.11. После месяца жалоб пользователей компания VeriSign удалила метазапись и вообще отключила эту службу. Не исключено, что после разбирательства в этом деле юристов служба вновь заработает. В конце концов, IETF может сузить спецификации, чтобы вообще запретить применение специальных записей.

В течение нескольких последних лет DNS претерпела ряд важных изменений. В частности, стандарты DNS, имеющие отношение к IPv6 и безопасности DNS, были существенно переработаны организацией IETF, вследствие чего все темы, посвященные этому вопросу в предыдущих изданиях этой книги, нужно переписывать заново. В табл. 15.4 перечислены основные изменения и сказано, где можно найти подробные сведения по тому или иному вопросу.

Некоторые из новых возможностей — это огромные проекты, еще не прошедшие стандартизацию в IETF. Рабочие группы, отвечающие за составление стандартов, имеют в своих рядах хороших писателей, но им недостает программистов, которые занимались бы тестированием спецификаций на постоянной основе. Поэтому некоторые из новых спецификаций очень трудно или вообще невозможно реализовать. В текущей версии пакета BIND (9.4) поддерживается большинство новых возможностей.

📖 Стандарт IPv6 был описан в главе 12.

Таблица 15.4. Недавние разработки в DNS и BIND

Страница	Номер документа RFC	Описание
388	3492	Интернационализированные доменные имена в кодировке Punycode.
389	2671	ENDS0 — изменения и расширения протокола.
394	1996	Асинхронные уведомления об изменениях в зоне.
400	2317	Бесклассовое делегирование в домене `in-addr.arpa` (специальное применение записи CNAME).
402	2782, 3958	Записи SRV, указывающие местонахождение служб.
404	—	Записи AAAA, указывающие адреса IPv6 (A6 уже не используется).
405	2672-3	Неконтролируемые записи DNAME.
405	—	`ip6.arpa` для обратных преобразований IPv6; неконтролируемые записи `ip6.int`.
—	3596, 3646	Поддержка IPv6.
447	1995	Инкрементные зонные пересылки.
448	2136	Динамические обновления (для организаций, в которых используется протокол DHCP).
453	2845, 2930, 3645	Спецификации TSIG/TKEY (подписи и обмен ключами в транзакции).
456	3225-6, 4033-5	Спецификация DNSSEC (аутентификация и безопасность данных зоны[a]).

[a] Полностью переделаны в 2004 г.

Два крупномасштабных дополнения — поддержка стандарта IPv6 и спецификация DNSSEC — заслуживают более подробных комментариев. В IPv6 длина IP-адреса возросла с 32 до 128 бит. Если бы этот стандарт был полностью реализован, он оказал бы огромное влияние на всю сеть Internet. Пакет BIND 9 поддерживает те компоненты IPv6, которые прошли процесс стандартизации, но маловероятно, чтобы на протяжении жизненного цикла этой книги стандарт IPv6 получил широкое распространение. Поэтому мы лишь вкратце затронем вопросы поддержки IPv6 в BIND 9.

Спецификация DNSSEC добавляет средства аутентификации к базе данных и серверам DNS. Она предписывает использовать алгоритм шифрования с открытым ключом для проверки источника данных, поступающих по протоколу DNS, и их целостности. В этой спецификации посредством DNS распространяются не только данные о компьютерах, но и ключи шифрования.

Описаны также более простые механизмы аутентификации, например системы совместного использования секретного ключа. Но такой ключ должен передаваться каждой паре серверов для взаимной аутентификации. Эта схема хорошо работает в небольшой организации, располагающей несколькими серверами, однако она неприменима в глобальном масштабе. BIND 9 реализует системы аутентификации DNSSEC (используется открытый ключ) и TSIG (используются сигнатуры транзакций в виде совместных секретных ключей).

Версии BIND, начиная с 9.3, содержат новые спецификации DNSSEC. Тем не менее, когда два года назад пользователи стали экспериментировать с подписанными зонами, то выяснилось, что исходная система DNSSEC не была практичной. В исходной систе-

ме родительская зона подписывала ключ дочерней зоны, а копии подписанного ключа хранились в обеих зонах. Если дочерней зоне нужно было изменить этот ключ, то ей приходилось связываться с родительской зоной и просить подписать новый ключ. Ну, здесь все было без проблем. А вот если родительской зоне нужно было изменить ключ, то она должна была обновлять все дочерние ключи, хранящиеся как в родительской, так и в дочерней зоне. В крупных зонах, таких как "com", эта операция выходила из-под контроля, поскольку в процессе обновления некоторые дочерние зоны постоянно оказывались вне досягаемости. Их ключи могли оказаться несинхронизированными, в результате чего у DNS не было никакой возможности проверить подписи.

Сейчас принята следующая схема: подписанный ключ хранится только в дочерней зоне, а родительская зона получает новую запись о ресурсах, называемую DS (delegation signer — подписывающая сторона, выполняющая делегирование). Технология DNSSEC будет рассмотрена в разделе 15.15.

Интернационализация доменных имен, позволяющая использовать символы других алфавитов, отличных от английского, выполняется за счет преобразования символов Unicode в символы ASCII. Система Punycode выполняет это преобразование уникально и с возможностью обратного преобразования с помощью алгоритма, известного под названием Bootstring (он описан в документе RFC3492). К 2005 г. регистраторы стали опубликовывать имена Punycode; некоторая форма этой системы поддерживается и во многих браузерах. К сожалению, в этой системе тоже есть свои ошибки, связанные с безопасностью и подменой IP-адресов (т.н. спуфинг, от *англ.* spoofing). Кроме того, интернационализированные имена доменов существенно сократили максимальную длину (как общую, так и отдельных компонентов) DNS-имен.

Нынешняя схема интернационализации позволила избавиться от ключевой проблемы — совмещения имен, с которой обычно справиться очень тяжело. Это стало возможным благодаря устранению несоответствий при выполнении преобразований между символами языков стран Азии и символами Unicode, зашифрованными с помощью Punycode, которые представляют их в DNS. Если один символ может иметь три, четыре или десять разных значений, будучи представленным в Unicode, то нужно руководствоваться стандартными правилами перевода, а символы, отображенные на экране монитора, должны быть предназначены для того, чтобы различать разные значения.

IPv6, DNSSEC и интернационализация существенно повысили размер записей о данных DNS, вследствие чего, естественно, DNS столкнется с ограничениями на размеры пакетов UDP.

В середине 90-х гг. протокол DNS был усовершенствован механизмами инкрементных зонных пересылок (напоминает применение команды `diff` для сравнения старого и нового зонных файлов; прототипом послужила программа `patch`, написанная Ларри Уоллом), асинхронных уведомлений (для информирования подчиненных серверов об изменении файлов главного сервера) и динамических обновлений (для DHCP-хостов). Все эти нововведения способствовали существенному расширению функциональных возможностей DNS, но они не решили фундаментальную транспортную проблему.

В конце 90-х гг. появилась спецификация EDNS0 (расширенный протокол DNS, версия 0), которая устраняла некоторые недостатки DNS. В этой спецификации запрашивающая сторона сообщает размер буфера сборки пакетов, поддерживаемые опции и версии протокола. Если сервер имен отвечает выдачей сообщения об ошибке, запрашивающая сторона возвращается к исходному протоколу DNS. Пакет BIND 9 реализует спецификацию EDNS0 как на стороне сервера, так и на стороне распознавателя.

15.7. База данных DNS

Доменная база данных представляет собой набор текстовых файлов, которые системный администратор ведет на главном сервере имен домена. Часто их называют файлами зон. В них содержатся элементы двух типов: директивы синтаксического анализатора (например, `$ORIGIN` и `$TTL`) и записи о ресурсах. Первые не являются частью базы данных, а предназначены для упрощения ввода записей.

В этом разделе мы сначала ознакомимся с форматами записей о ресурсах, которые регламентируются документами RFC1035, RFC1183, RFC1876, RFC2230, RFC2782, RFC2930, RFC3596 и RFC3658. Описание директив приведено в разделе 15.7.

Записи о ресурсах

С каждой зоной в иерархии DNS связан набор записей о ресурсах. Базовый формат записи таков:

```
[имя] [ttl] [класс] тип данные
```

Поля разделяются знаками табуляции или пробелами и могут содержать специальные символы (табл. 15.5).

Таблица 15.5. Специальные символы, используемые в записях о ресурсах

Символ	Назначение
`;`	Начало комментария
`@`	Имя текущей зоны
`()`	Разбивка данных на несколько строк
`*`	Метасимвол[a] (только в поле *имя*)

[a] Некоторые примеры можно найти в разделе 15.7.

Поле *имя* идентифицирует объект (обычно хост или домен), к которому относится запись. Если несколько последовательно расположенных записей ссылаются на один и тот же объект, то после первой записи, а также всех последующих записей, начинающихся с пробела, поле *имя* можно опустить. Поле *имя* должно начинаться в первой колонке, если оно присутствует.

Имя может быть сокращенным либо полным. Имена второго типа заканчиваются точкой и полностью определены. На внутреннем уровне программное обеспечение работает с полными именами, добавляя имя текущего домена и точку ко всем именам, которые не заканчиваются точкой. Это позволяет укорачивать имена, но часто сопряжено с ошибками.

Например, в домене `cs.colorado.edu` имя `anchor` интерпретируется как “anchor.cs.colorado.edu.”. Если же ввести имя `anchor.cs.colorado.edu`, то отсутствие точки в конце также будет подразумевать сокращенное имя, к которому следует добавить стандартный домен, что в итоге даст имя “anchor.cs.colorado.edu.cs.colorado.edu.”. Это очень распространенная ошибка.

В поле *ttl* (Time To Live — время жизни) задается время (в секундах), в течение которого элемент данных может оставаться в кэше и при этом считаться достоверным. Это поле часто опускают, но оно обязательно присутствует в файле, содержащем корневые “подсказки”. Стандартное значение поля задается директивой `$TTL`, указываемой в начале зонного файла. В BIND 9 эта директива обязательна. В BIND 8, если она отсутст-

вует, поле *ttl* определяется отдельно для каждой зоны на основании значения, задаваемого в записи SOA.

Если время жизни записей задать равным примерно неделе, то это приведет к значительному снижению сетевого трафика и нагрузки на DNS. Однако следует помнить о том, что после сохранения записи в кэше за пределами локальной сети ее уже нельзя принудительно сделать недостоверной. Поэтому, планируя провести реструктуризацию сети, сделайте значение `$TTL` достаточно низким (например, один час), чтобы записи, находящиеся во внешних кэшах, быстро устарели.

📖 В главе 17 вы найдете подробные сведения о NIS.

В некоторых сайтах устанавливают малое время жизни в записях Internet-серверов. Это делается для того, чтобы в случае возникновения проблем на сервере (сбой в сети, сбой в оборудовании, атака типа "отказ в обслуживании" и т.п.) администратор мог среагировать, изменив преобразование имени в IP-адрес. Так как исходные значения времени жизни были очень малыми, новые значения будут распространяться очень быстро. Например, имя `google.com` имеет пятиминутное время жизни, а у серверов имен Google время жизни составляет четыре дня (345 600 секунд):

```
google.com.          300      IN   A    216.239.37.99
google.com.          345600   IN   NS   ns1.google.com
ns1.google.com.      345600   IN   A    216.239.32.10
```

Для получения этих данных мы использовали команду `dig` (`dig@ns1.google.com google.com`), а полученные результаты немного сократили.

BIND 9 реализует концепцию, известную как гармонизация времени жизни, в соответствии с которой все записи в RRset (то есть, все однотипные записи, принадлежащие одному хосту) имеют одинаковое время жизни. Используемое значение соответствует значению первой записи ресурса для пары хост/тип.

В поле *класс* задается тип сети. Распознаются три значения:

- `IN` — Internet;
- `CH` — ChaosNet;
- `HS` — Hesiod.

По умолчанию в этом поле задается значение `IN`. Оно часто задается явным образом в файлах данных зоны, несмотря на то, что оно используется по умолчанию. Hesiod (разработка MIT — Массачусетсского технологического института) представляет собой службу базы данных — по сути, это надстройка пакета BIND. ChaosNet — это почти исчезнувший сетевой протокол, ранее применявшийся на Lisp-машинах компании Symbolics.

На сегодняшний день только две порции идентификационных данных обычно скрыты в классе ChaosNet: номер версии ПО сервера имен и имя компьютера, на котором работает этот сервер. Эти крупицы данных можно извлекать с помощью команды `dlg`, о которой будет рассказано в разделе 15.8. Администраторы пользуются номером версии сервера имен для того, чтобы идентифицировать серверы во время модернизации; идентификацию хостов они используют для отладки серверов имен, дублированных в процессе использования альтернативной маршрутизации (anycast routing). Раньше эта информация была доступна в классе CH благодаря особенности реализации BIND; теперь для нее выработан стандарт (этим занимается IETF) как часть DNS[5].

[5] К сожалению, сейчас нет единого мнения относительно того, каким должно быть имя для файла данных. Предлагаются следующие форматы: *версия*`.bind`, *имя_компьютера*`.bind`, `id-`*сервер*, и т.п.

Существуют различные типы DNS-записей, из которых широко используются менее десяти. В стандарте IPv6 добавилось еще несколько типов. Записи о ресурсах разбиваются на четыре группы:

- зонные записи — определяют домены и их серверы имен;
- базовые записи — связывают имена с адресами и обеспечивают маршрутизацию электронной почты;
- аутентификационные записи — предоставляют информацию, касающуюся аутентификации и сигнатур;
- вспомогательные записи — содержат дополнительную информацию о компьютерах и доменах.

Содержимое поля *данные* зависит от типа записи (табл. 15.6).

Таблица 15.6. Записи базы данных DNS

	Тип	Название	Назначение/содержимое
Зонные	SOA	Start Of Authority — начало полномочий	Определение DNS-зоны
	NS	Name Server — сервер имен	Определение серверов имен зоны, делегирование полномочий поддоменам
Базовые	A	IPv4 Address — адрес IPv4	Преобразование имени в адрес IPv4
	AAAA[a]	IPv6 Address — адрес IPv6	Преобразование имени в адрес IPv6
	PTR	Pointer— Указатель	Преобразование адреса в имя
	MX	Mail Exchanger — обмен почтой	Маршрутизация электронной почты
Аутентификационные	DS	Delegation Signer — подписывающая сторона, выполняющая делегирование	Хеш ключа подписания подписанной дочерней зоны
	DNSKEY	Public Key — Открытый ключ	Открытый ключ для DNS-имени
	NSEC	Next Secure — следующая защита	Используется наряду с DNSSEC для отрицательных ответов
	RRSIG	Signature — Сигнатура	Набор подписанных аутентифицированных записей о ресурсах
Вспомогательные	CNAME	Canonical Name — Каноническое имя	Дополнительные имена (псевдонимы) хоста
	LOC	Location — местоположение	Географическое местоположение и физический размер DNS-объекта
	SRV	Services — службы	Местонахождение известных служб в пределах домена
	TXT	Text — текст	Комментарии или нетипизированная информация

[a] Адресные записи AAAA и A6 IPv6 являются, если можно так выразиться, “спарринг-партнерами” в IETF в течение последних нескольких лет. AAAA в конце концов выиграла состязание и стала стандартом, хотя одно время сичталась устаревшей. A6 сейчас является экспериментальной.

Есть и другие типы записей, которые либо устарели, либо являются экспериментальными, либо не нашли широкого применения. Полный их список приведен в документации к пакету BIND. Большинство записей ведется вручную (посредством редактирования текстовых файлов), хотя записи о ресурсах защиты требуют криптографической

обработки (шифрования) и, поэтому, должны обрабатываться программными средствами. Эти записи описаны в подразделе DNSSEC раздела 15.15.

Порядок записей о ресурсах почти произволен. Запись SOA раньше должна была быть первой, однако сейчас это требование было смягчено. Обычно сразу же после записи SOA стоят записи NS. Записи по каждому хосту, как правило, группируются. Вообще это довольно распространенная практика — сортировать записи по полю *имя*, хотя на некоторых сайтах сортировка выполняется по IP-адресам, чтобы было легче идентифицировать неиспользуемые адреса.

Подробно описывая все типы записей о ресурсах, мы в качестве примера рассмотрим записи из базы данных домена cs.colorado.edu. Поскольку в данном случае по умолчанию выбран домен "cs.colorado.edu.", то имя хоста `anchor` в действительности означает "anchor.cs.colorado.edu.".

Запись SOA

Запись SOA (start of authority — начало полномочий) обозначает начало *зоны* — группы записей о ресурсах, расположенных в одной точке пространства имен DNS. Этот хост дерева DNS называют также точкой передачи полномочий. Как мы увидим ниже, домен DNS обычно охватывает минимум две зоны: одна служит для преобразования имен компьютеров в IP-адреса, а остальные — для обратного преобразования. Та часть дерева DNS, которая служит целям прямого преобразования, упорядочена по именам, а ветвь обратного преобразования — по IP-адресам.

Для каждой зоны создается всего одна запись SOA. Запись SOA содержит имя зоны, контактный адрес администрации зоны, порядковый номер и различные параметры обновления данных. Например:

```
; Начало зоны для домена cs.colorado.edu

@    IN  SOA  ns.cs.colorado.edu. hostmaster.cs.colorado.edu. (
                2001111300    ; Порядковый номер
                7200          ; Период обновления          (2 часа)
                1800          ; Интервал между попытками   (30 минут)
                604800        ; Интервал устаревания       (1 неделя)
                7200 )        ; Минимальное время жизни    (2 часа)
```

Поле *имя* содержит здесь символ `@`, обозначающий имя текущей зоны. В данном примере можно было вместо него использовать "cs.colorado.edu.". Текущее имя задается в инструкции `zone` конфигурационного файла сервера имен; его можно изменить в зонном файле посредством директивы `$ORIGIN` (описана в разделе 15.7).

В показанном фрагменте поле *ttl* отсутствует. Класс зоны — IN (Internet), тип записи — SOA, а остальные элементы составляют поле *данные*.

Имя "`ns.cs.colorado.edu.`" указывает на главный сервер имен этой зоны.

Имя "`hostmaster.cs.colorado.edu.`" определяет адрес электронной почты для контактов с администратором домена. Адрес дан в формате "*пользователь*`.`*хост*`.`" (а не *пользователь*`@`*хост*). Если необходимо отправить сообщение администратору, просто замените первую точку символом `@` и уберите хвостовую точку. Вместо реального регистрационного имени часто используется псевдоним, например `admin` или `hostmaster`. Дело в том, что на должность администратора может заступить другой человек. В таком случае проще поменять одну запись в файле `aliases` (описывается в разделе 19.4), чем выискивать все упоминания старого адреса в зонных файлах.

Круглые скобки позволяют разбить запись SOA на несколько строк. Их размещение не произвольно в BIND 4 и BIND 8: мы попробовали укоротить первую строку, поставив открывающую скобку перед контактным адресом, но BIND не смогла распознать такую запись. В некоторых реализациях круглые скобки распознаются только в записях SOA и TXT. В BIND 9 используется улучшенный модуль синтаксического анализа, поэтому круглые скобки могут стоять где угодно.

Первый числовой параметр — это порядковый номер конфигурации зоны. С его помощью подчиненные серверы определяют, когда следует загружать новые данные. Порядковым номером может быть любое 32-битное целое число, причем оно должно увеличиваться при каждом изменении зонного файла. Во многих организациях в этом номере зашифровывается дата последней модификации файла. Например, значение 2004111300 указывает на первое изменение в зонном файле, сделанное 13 ноября 2004 г.

Порядковые номера могут не быть непрерывными, но должны монотонно возрастать. Если случайно задать на главном сервере очень большое число и передать его подчиненным серверам, то исправить порядковый номер на главном сервере не удастся. Подчиненные серверы запрашивают новые данные только в том случае, когда порядковый номер записи SOA главного сервера больше, чем у них.

Существуют три способа решения этой проблемы; только первые два применимы к BIND 9.

- Первый способ заключается в использовании особенностей пространства последовательностей, из которого выбираются порядковые номера. Суть в том, что к имеющемуся номеру добавляется "магическое" число (2^{31}), заставляющее подчиненные серверы обновить свои данные, после чего устанавливается требуемый номер. Подробнее об этом рассказывается в документе RFC1982
- Более утомительный способ — изменить порядковый номер на главном сервере, уничтожить подчиненные серверы, удалить их резервные базы данных, чтобы заставить их выполнить перезагрузку из главного сервера, а затем перезапустить серверы. Простое удаление и файлов и перезагрузка не помогут; вы должны будете уничтожить и перезапустить подчиненные серверы.
- В BIND 4.9 и BIND 8 есть прием, позволяющий сбросить порядковый номер в нуль на один интервал обновления, а затем заново начать нумерацию. Нулевой номер всегда вызывает обновление, поэтому не забудьте присвоить ему нормальное значение, после того как все подчиненные серверы загрузят зонные данные.

Часто пользователи делают одну и ту же ошибку: меняют файлы данных, забывая при этом исправить порядковый номер. "В наказание" ваш сервер имен откажется распространить внесенные изменения на подчиненные серверы.

Следующие четыре элемента записи SOA — значения интервалов времени (по умолчанию в секундах), определяющих, как долго данные могут находиться в кэше в различных точках глобальной базы данных DNS. Можно поменять единицы измерения, воспользовавшись суффиксами `m` (минуты), `h` (часы), `d` (дни) и `w` (недели). Например, выражение `1h30m` означает "1 час 30 минут". Выбор интервалов времени требует компромисса между эффективностью (использование старого значения дешевле выборки нового) и точностью (новые значения более точны).

Ниже показана копия вышеприведенного примера записи SOA, чтобы вам не нужно было открывать предыдущую страницу:

```
; Начало зоны для домена cs.colorado.edu
@    IN  SOA  ns.cs.colorado.edu. hostmaster.cs.colorado.edu. (
               2001111300    ; Порядковый номер
               7200          ; Период обновления
               1800          ; Интервал между попытками
               604800        ; Интервал устаревания
               7200 )        ; Минимальное время жизни
```

Первый элемент задает периодичность обновления данных. Он показывает, как часто подчиненные серверы должны связываться с главным сервером и проверять, не поменялся ли порядковый номер конфигурации зоны. Если зонная база данных изменилась (порядковый номер главного сервера стал *больше*, чем у подчиненного сервера), подчиненные серверы должны обновить свои копии базы данных. Общепринятые значения для этого интервала — от одного до шести часов (3 600–21 600 секунд).

Вместо того чтобы пассивно ждать истечения периода обновления, современные серверы BIND самостоятельно уведомляют свои подчиненные серверы об изменениях зон, если только в конфигурационном файле не отключен параметр `notify`. Серверы, понимающие подобные уведомления, немедленно обновляют свои базы данных. Не исключено, что уведомление об обновлении может быть потеряно из-за перегрузки в сети, поэтому периоду обновления нужно всегда присваивать приемлемое значение.

Если подчиненный сервер пытается узнать порядковый номер у главного сервера, а тот не отвечает, то через некоторое время будет сделана повторная попытка. Наш опыт показывает, что нормальное значение для второго элемента — от 20 до 60 минут (1 200–3 600 секунд).

Если главный сервер длительное время отключен, то подчиненные серверы будут безуспешно пытаться обновить свои данные. По истечении определенного времени все подчиненные серверы должны “решить”, что главный сервер не включится никогда и его данные наверняка устарели. Интервал устаревания определяет, как долго подчиненные серверы будут обслуживать домен в отсутствие главного сервера. Система должна просуществовать, даже если главный сервер не работает неделю, поэтому третьему элементу следует присваивать большое значение. Мы рекомендуем выбирать интервал от недели до месяца.

Четвертый элемент определяет время нахождения в кэше отрицательных ответов[6]. Время жизни положительных ответов (т.е. собственно записей) устанавливается директивой `$TTL` в начале зонного файла. Как свидетельствует практика, значение `$TTL` должно быть от нескольких часов до нескольких дней, а минимальное время жизни отрицательных ответов — один–два часа (но не более трех). Значение `$TTL` должно быть больше или равно минимальному времени жизни.

Значение `$TTL`, интервал устаревания и минимальное время жизни в конечном итоге заставляют всех пользователей DNS удалять старые данные. Изначально система доменных имен основывалась на том, что информация об хостах относительно стабильна и меняется нечасто. Но с появлением протокола DHCP и портативных компьютеров все перевернулось. Теперь разработчики пакета BIND отчаянно пытаются угнаться за временем, внедряя механизмы инкрементных зонных пересылок и динамических обновлений (описаны в разделе 15.14). О директиве `$TTL` и концепции согласования времени жизни рассказывается в разделе 15.7.

[6] До появления BIND 8.2 четвертый элемент задавал стандартное время жизни записей о ресурсах. Он включался в каждую запись и использовался для проверки устаревания кэшированных записей на неавторитетных серверах.

Записи NS

Записи NS (name server — сервер имен) идентифицируют серверы имен, которые авторитетны для зоны (т.е. все главные и подчиненные серверы), а также делегируют полномочия по управлению поддоменами другим организациям. Обычно эти записи стоят после записи SOA. Их формат таков:

```
зона [ttl] IN NS имя_хоста
```

Например:

```
cs.colorado.edu.    IN  NS   ns.cs.colorado.edu.
cs.colorado.edu.    IN  NS   anchor.cs.colorado.edu.
cs.colorado.edu.    IN  NS   nc.cs.utah.edu.
```

Если имя зоны совпадает с полем *имя* предыдущей записи SOA, то поле *зона* можно оставить пустым. Таким образом, строки

```
                    IN  NS   ns.cs.colorado.edu.
                    IN  NS   anchor.cs.colorado.edu.
                    IN  NS   nc.cs.utah.edu.
```

стоящие после записи SOA для зоны `cs.colorado.edu`, будут эквивалентны приведенным выше.

Следует указывать все авторитетные серверы имен домена `cs.colorado.edu`, которые должны быть видимы внешнему миру, причем это необходимо делать как в зонном файле домена, так и в файле родительской зоны `colorado.edu`. Кэширующие серверы не могут быть авторитетными, поэтому их приводить не нужно. В записи NS нет ключевого слова, которое определяло бы тип сервера (главный или подчиненный). Эта информация находится в конфигурационном файле сервера имен.

На основании записей NS, относящихся к конкретной зоне, BIND идентифицирует подчиненные серверы, когда ему нужно разослать уведомления об изменении зоны. Те же самые записи, присутствующие в описании родительской зоны (`colorado.edu`), определяют поддомен “cs” и передают полномочия по его управлению соответствующим серверам имен. Если список серверов имен в описании родительской зоны не совпадает с аналогичным списком самой зоны, то любой новый сервер становится невидимкой и не используется для ответов на запросы из внешнего мира. Такая конфигурация иногда встречается вследствие ошибок проектирования, иногда — по причине забывчивости. При этом ничего страшного не случится, если в родительской зоне будет хотя бы одна действительная запись NS для дочерней зоны.

Беглый взгляд на информацию о делегировании в нашей зоне позволил обнаружить крупный сервер домена `colorado.edu`, о котором домен “edu” ничего не “знал”. Стоит иногда с помощью утилиты `dig` проверять информацию о делегировании, чтобы убедиться в наличии правильного набора серверов (см. раздел 15.16).

О делегировании можно прочитать в разделе 15.7.

Записи A

Записи A (address — адрес) являются основной частью базы данных DNS. Они обеспечивают перевод имен компьютеров в IP-адреса (ранее эта информация хранилась в файле `/etc/hosts`). Для каждого из сетевых интерфейсов компьютера обычно существует одна запись A. Формат ее таков:

```
имя_хоста [ttl] IN A IP-адрес
```

Например:

```
anchor       IN  A   128.138.243.100
```

Если компьютер оснащен несколькими сетевыми интерфейсами, то с ними можно связать общее доменное имя или закрепить за каждым свое имя.

Записи PTR

Записи PTR (pointer — указатель) обеспечивают обратный перевод IP-адресов в доменные имена. Как и в случае записей A, для каждого сетевого интерфейса компьютера должна существовать одна запись PTR. Но прежде чем знакомиться с этими записями, давайте немного отвлечемся и поговорим о специальном домене верхнего уровня, который называется `in-addr.arpa`.

Полностью определенные доменные имена можно рассматривать как записи, в которых наиболее важная часть стоит справа. Например, в имени `anchor.cs.colorado.edu` хост `anchor` принадлежит домену "cs", последний находится в домене "colorado", а тот — в домене "edu". С другой стороны, в IP-адресах самая важная часть стоит слева. В адресе 128.138.243.100 хост 100 находится в подсети 243, которая является частью сети 128.138.

Домен `in-addr.arpa` был создан для того, чтобы при прямых и обратных преобразованиях использовался один и тот же набор программных модулей и единое дерево имен. Домены в ветви `in-addr.arpa` именуются как IP-адреса с обратным порядком следования байтов. Например, зона для нашей подсети 243 называется `243.138.128.in-addr.arpa`.

Общий формат записи PTR таков:

```
адрес [ttl] IN PTR имя_хоста
```

Запись PTR в зоне `243.138.128.in-addr.arpa`, соответствующая приведенной выше записи A для хоста `anchor`, будет иметь следующий вид:

```
100        IN  PTR   anchor.cs.colorado.edu.
```

Имя 100 не заканчивается точкой и потому является сокращенным. Вопрос: относительно чего? Ни в коем случае не относительно домена "cs.colorado.edu.". Для того чтобы эта запись была точной, домен по умолчанию должен называться "243.138.128.in-addr.arpa.".

Этого можно добиться, поместив записи PTR для каждой подсети в отдельный файл, за которым в конфигурационном файле сервера имен закрепляется стандартный домен. Другой способ выполнить обратное преобразование — включить в зонный файл записи вида

```
100.243    IN  PTR   anchor.cs.colorado.edu.
```

с доменом по умолчанию `138.128.in-addr.arpa`. В некоторых организациях все записи обратного преобразования помещаются в один файл, а подсеть задается посредством директивы `$ORIGIN`. Обратите внимание на то, что имя `anchor.cs.colorado.edu` должно оканчиваться точкой, иначе к нему будет добавлена строка 138.128.in-addr.arpa.

Поскольку `cs.colorado.edu` и `243.138.128.in-addr.arpa` — разные области пространства имен DNS, они образуют две отдельные зоны. У каждой из этих зон должна быть своя запись SOA и свои записи о ресурсах. Помимо определения зоны `in-addr.arpa` для каждой реальной сети нужно также задать зону, которая охватывала бы интерфейс обратной связи, 127.0.0.0.

Описанные привязки прекрасно работают, когда маски подсетей проходят по границе байтов. Но как выполнять обратные преобразования для подсети вида 128.138.243.0/26?

В документе RFC2317 описан изящный прием, основанный на применении записей CNAME; подробнее об этом речь пойдет в разделе 15.7.

Обратные преобразования, выполняемые записями PTR, используются всеми программами, которые аутентифицируют входящий сетевой трафик. Например, демон `sshd` может допускать удаленную регистрацию без пароля, если исходный хост указан по имени в пользовательском файле `~/.shosts`. Когда хост-адресат получает запрос на установление соединения, он "знает" хост-отправитель только по IP-адресу. Пользуясь услугами DNS, он преобразует IP-адрес в имя, которое сравнивается с содержимым соответствующего файла. Программы `netstat`, `tcpd`, `sendmail`, `sshd` и `ftpd` — все они получают имена компьютеров из IP-адресов с помощью механизма обратного преобразования.

Важно, чтобы записи A соответствовали записям PTR. Несовпадение и отсутствие последних приводит к ошибкам аутентификации, в результате чего система замедляет работу. Это само по себе неприятно, но еще хуже то, что появляется почва для атак типа "отказ в обслуживании", которые направлены на любое приложение, требующее соответствия записям A при обратном преобразовании.

Записи MX

Записи MX (mail exchanger — обмен почтой) используются системой электронной почты для более эффективной маршрутизации почтовых сообщений. Запись MX подменяет адресата сообщения, в большинстве случаев направляя сообщение концентратору электронной почты на сервере получателя, а не прямо на его рабочую станцию.

Запись MX имеет следующий формат:

имя [*ttl*] IN MX *приоритет хост* ...

Ниже показаны два примера: один — для хоста, который самостоятельно получает адресованную ему почту, если только не выключен, а второй — для хоста, пользователи которого вынуждены загружать почту с сервера.

```
piper       IN  MX   10 piper
            IN  MX   20 mailhub
            IN  MX   50 boulder.colorado.edu.
xterm1      IN  MX   10 mailhub
            IN  MX   20 anchor
            IN  MX   50 boulder.colorado.edu.
```

Сначала опрашиваются хосты с низким приоритетом (наиболее предпочтительный приоритет — 0, самый нежелательный — 65535). Например, почта, адресованная пользователю `bob@xterm1`, будет маршрутизироваться следующим образом. Если доступен концентратор `mailhub`, почта посылается туда; в противном случае она посылается на хост `anchor`. Если оба отключены, почта направляется на хост `boulder`. При этом имя хоста `boulder` должно быть полностью определенным, поскольку он не является членом стандартного домена (в данном случае — `cs.colorado.edu.`).

Список приоритетов и хостов можно давать в одну строку, но отдельные строки легче читать. Между значениями приоритетов оставляйте числовое пространство, чтобы в случае необходимости всегда можно было добавить нового адресата.

Записи MX полезны во многих ситуациях, например:

- если в системе есть центральный концентратор почты;
- если хост-адресат выключен;
- если адресат не подключен к сети Internet;

- если хост-адресат не понимает протокол SMTP;
- если администратор локальной сети лучше знает, куда следует рассылать сообщения.

В первом из этих случаев почта направляется на концентратор — компьютер, где пользователи обычно читают почту. Во втором случае почта переадресуется на ближайший компьютер и возвращается на хост-адресат, когда тот вновь включится.

Если компьютер не подключен к Internet, у него не может быть записи A в DNS, но могут быть записи MX. Компьютеры, имеющие только такие записи, либо скрыты за брандмауэром, либо обслуживаются Internet-провайдером, либо большую часть времени выключены. Программа `sendmail` не устанавливает непосредственное соединение с таким адресатом, но может переслать почту поближе к нему, связавшись с одним из хостов, которые указаны в записях MX.

Записи MX могут применяться еще и потому, что администратор локальной сети гораздо лучше знает организацию почтовой системы, чем те, кто присылает письма. За администратором остается право определять, по каким каналам распространяются почтовые потоки.

Каждый хост, о котором известно во внешнем мире, должен иметь записи MX. Для второстепенных хостов достаточно одного–двух альтернативных вариантов. Крупный хост должен иметь несколько записей:

- одну для себя как основной вариант;
- одну для локального почтового концентратора как второй вариант;
- одну для центрального концентратора домена или его родительского домена как резервный вариант.

У самого домена должна быть запись MX для хоста-концентратора, чтобы можно было посылать почту по схеме *пользователь@домен*. Естественно, такая конфигурация требует наличия уникальных пользовательских имен в пределах всех компьютеров домена. Например, чтобы иметь возможность посылать почту пользователю `evi@cs.colorado.edu`, нужны либо хост `cs`, либо записи MX в домене `cs.colorado.edu` (возможно, и то и другое):

```
cs          IN  MX   10 mailhub.cs.colorado.edu.
            IN  MX   20 anchor.cs.colorado.edu.
            IN  MX   50 boulder.colorado.edu.
```

Компьютер, принимающий почту для другого хоста, должен содержать его имя в файлах конфигурации `sendmail`. В разделе 18.9 будет рассказываться о средстве `use_cw_file` и файле `local-host-names`, которые применяются в `sendmail` для этих целей.

В базе данных DNS иногда можно встретить метазаписи MX:

```
*           IN  MX   10 mailhub.cs.colorado.edu.
```

На первый взгляд кажется, что такая запись позволяет избежать многократного ввода данных и является стандартной записью для всех хостов. Однако метасимвол интерпретируется совсем не так, как можно ожидать. Он соответствует полю имени, которое еще *не* было указано в явном виде в других записях о ресурсах.

Следовательно, нельзя использовать звездочку с целью задания стандартного значения для всех своих компьютеров. Можно — что неправильно — задать посредством нее маршрут для “чужих” компьютеров. В результате на концентратор будет поступать масса почтовых сообщений, тут же отвергаемых по той причине, что имя хоста, обозначенное

звездочкой, не принадлежит домену. Поэтому необходимо избегать применения метасимвола в записях MX.

Записи CNAME

Записи CNAME (canonical name — каноническое имя) позволяют назначать хосту дополнительные мнемонические имена. Псевдонимы широко применяются для закрепления за компьютером какой-либо функции либо просто для сокращения его имени. Реальное имя иногда называют *каноническим*. Приведем несколько примеров:

```
ftp         IN  CNAME     anchor
kb          IN  CNAME     kibblesnbits
```

Запись CNAME имеет следующий формат:

псевдоним [*ttl*] IN CNAME *имя_хоста*

Обнаружив запись CNAME, демон `named` перестает посылать запросы по мнемоническому имени и переключается на реальное имя. Если у компьютера есть запись CNAME, то другие записи для хоста (A, MX, NS и др.) должны ссылаться на его реальное, а не мнемоническое имя[7]. Например, строки

```
colo-gw     IN  A         128.138.243.25
moogie      IN  CNAME     colo-gw
www         IN  CNAME     moogie
```

являются правильными. Но если связать адрес или почтовый приоритет (посредством записей A и MX) с именем `www` или `moogie`, то это будет неверно.

В BIND длина цепочки записей CNAME не должна быть больше восьми. Цепочка — это ситуация, когда одна запись CNAME ссылается на другую, та — на третью и т.д. Последним, восьмым элементом должна быть реальная запись A.

Как правило, можно вообще отказаться от записей CNAME, используя записи A для настоящего имени и мнемонического имени хоста.

Специальное применение записей CNAME

С помощью записей CNAME можно организовать поддержку зон обратного преобразования для сетей, где маски подсетей не проходят по границе байтов. До того как протокол CIDR получил широкое распространение, такая организация подсетей не применялась или, по крайней мере, "неправильные" подсети существовали в рамках одной организации, поэтому управлять зонами обратного преобразования было несложно. Например, если сеть 128.138 класса B разделить на группу подсетей класса C, то каждая подсеть займет строго оговоренное место в домене `in-addr.arpa`. В частности, зоной обратного преобразования для подсети 243 будет `243.138.128.in-addr.arpa`.

О протоколе CIDR рассказывалось в разделе 12.3.

Но что произойдет, если подсеть 243 разделить еще, допустим, на четыре подсети /26? Если все они находятся в пределах одной организации, то это — не проблема: они попрежнему описываются в одном файле, содержащем все их записи PTR. Однако может случиться так, что подсеть 243 принадлежит провайдеру Internet, который делегирует под-

[7] Это правило для записей CNAME было явно ослаблено для DNSSEC, где цифровые сигнатуры добавляются к каждому набору записей о ресурсах. Запись RRSIG для CNAME ссылается на мнемоническое имя.

сети /26 разным клиентам. В этом случае требуется более сложное решение. Провайдер должен либо управлять записями зон обратного преобразования на стороне каждого клиента, либо найти способ разделить третий байт IP-адреса (в рассматриваемом примере 243) на четыре разных элемента, каждый из которых делегируется независимо.

Когда административная граница проходит не по границе байтов, приходится быть изворотливым. Нужно также тесно взаимодействовать с администрацией домена, находящегося выше или ниже в иерархии. Трюк заключается в следующем: для каждого возможного адреса хоста в зоне in-addr.arpa следует добавить запись CNAME, которая переводит операцию поиска в зону, управляемую владельцем соответствующей подсети. Подобная схема приводит к образованию громоздких зонных файлов в родительском домене, но она же позволяет передавать полномочия реальным пользователям каждой подсети.

Рассмотрим описываемый процесс подробнее. Родительская организация (в нашем случае — провайдер) создает для каждого возможного IP-адреса записи CNAME со специальным искусственным компонентом (отделяется точкой), представляющим подсеть. Например, для первого из четырех блоков адресов подсети /26 дополнительный компонент будет называться “0-63”, для второго — “64-127” и т.д. Вот как это выглядит:

```
$ORIGIN 243.138.128.in-addr.arpa.
1          IN  CNAME   1.0-63
2          IN  CNAME   2.0-63
...
63         IN  CNAME   63.0-63
64         IN  CNAME   64.64-127
65         IN  CNAME   65.65-127
...
```

Чтобы передать управление адресами 0-63 зоны обратного преобразования клиенту, за которым закреплена эта подсеть, нужно добавить следующие записи NS:

```
0-63       IN  NS     ns1.customer1.com.
0-63       IN  NS     ns2.customer1.com.
```

На хосте `customer1.com` будет зонный файл, содержащий привязки для зоны обратного преобразования `0-63.243.138.128.in-addr.arpa`. Например:

```
1          IN  PTR    host1.customer1.com.
2          IN  PTR    host2.customer1.com.
```

Добавив дополнительный компонент, мы создали новую точку передачи полномочий. Когда, например, кто-то ищет доменное имя, соответствующее адресу `128.138.243.1`, запись CNAME в точке `1.243.138.128.in-addr.arpa` перенаправит поиск в точку `1.0-63.243.138.128.in-addr.arpa`, а этим именем уже управляет реальный клиент.

Клиентские зонные файлы не содержат ничего лишнего: со всеми конфигурационными записями приходится иметь дело провайдеру. Ситуация усложняется, если клиент сам является провайдером и дальше делит свое адресное пространство. Однако непреодолимых препятствий не существует: в BIND поддерживаются цепочки записей CNAME длиной до восьми элементов, а поскольку в байте, как известно, восемь битов, превышения допустимого предела не произойдет. Различные документы RFC не одобряют, но и не запрещают применять такие цепочки. К негативным последствиям можно отнести то, что скорость распознавания имен замедляется, так как распознаватель должен пройти по каждому звену цепочки, послав соответствующее число запросов серверу.

Когда схема стала получать распространение, набор команд, понимаемых BIND, пополнился директивой `$GENERATE` (о ней будет идти речь в разделе 15.7), которая упрощает создание записей о ресурсах в родительской зоне.

Например, чтобы сгенерировать все необходимые записи для первой подсети, достаточно следующих строк:

```
$ORIGIN 243.138.128.in-addr.arpa.
$GENERATE 0-63 $ CNAME $.0-63
0-63       IN  NS     ns1.customer1.com.
0-63       IN  NS     ns2.customer1.com.
```

Метасимвол `$` в директиве `$GENERATE` является счетчиком цикла и приводит к созданию 64 различных записей CNAME. Остальные три подсети /26 обрабатываются аналогично.

Записи LOC

Запись LOC определяет географическое местоположение и (необязательно) физический размер (диаметр) объекта DNS. В настоящее время записи LOC не влияют на функционирование сети Internet, и существующие стандартные программы не выполняют их поиск. Но было предложено множество интересных применений информации, содержащейся в этих записях, включая трассировку и оптимизацию маршрутов, автоматические адресные привязки, а также сетевые исследования.

📖 Записи LOC описаны в документе RFC1819.

Формат записи LOC таков:

имя [*ttl*] IN LOC *широта долгота* [*высота* [*размер* [*гориз_точн* [*верт_точн*]]]]

Широта и долгота указываются в градусах, минутах и секундах (разделяются пробелами), после которых следует обозначение `N` (north — север), `S` (south — юг), `E` (east — восток) или `W` (west — запад). Секунды можно опустить. Допускается также указывать только градусы.

Последующие поля задаются в сантиметрах (по умолчанию) или в метрах (суффикс `m`). Аргумент *высота* — это высота объекта над уровнем моря; *размер* — диаметр воображаемой сферы, окружающей объект; *гориз_точн* — горизонтальная точность измерения, а *верт_точн* — вертикальная точность. По умолчанию размер равен одному метру, горизонтальная точность составляет 10 метров, а вертикальная — 10 километров.

Вот пример для хоста caida.org, расположенного в Сан-Диего, штат Калифорния:

```
caida.org.  IN  LOC  32 53 01 N 117 14 25 W 107m 30m 18m 15m
```

Многие утилиты визуализации, написанные организацией CAIDA (Cooperative Association for Internet Data Analysis — Совместная ассоциация по анализу данных в сети Internet), требуют наличия данных о широте и долготе, поэтому администраторам Web-сайтов рекомендуется включать соответствующую информацию в базы данных DNS. Но не все заинтересованы в том, чтобы кто угодно знал их местонахождение. В таких ситуациях можно использовать неточные значения. Они все равно представляют определенную ценность для людей, занимающихся сетевым анализом, и в то же время обеспечивают некоторую анонимность.

Записи SRV

Записи SRV определяют местонахождение служб в пределах домена. К примеру, благодаря этой записи можно запросить удаленный домен и узнать имя его FTP-сервера. Раньше в подобной ситуации приходилось действовать наугад в надежде на то, что администратор удаленного домена, следуя традиционной практике, добавил запись CNAME для имени "ftp" в базу данных DNS.

Гораздо разумнее применять для этих целей записи SRV. С их помощью администраторам намного удобнее менять адреса служб и контролировать их использование. Однако требуется, чтобы сами клиенты знали, как найти и проанализировать записи SRV, поэтому эффект от их применения пока не столь ощутимый. Эти записи широко используются в Windows.

Записи SRV напоминают обобщенные записи MX с дополнительными полями, которые позволяют администратору DNS управлять внешними соединениями и распределять нагрузку на сервер. Формат записей таков:

служба.*протокол*.*имя* [*ttl*] IN SRV *приоритет вес порт сервер*

Аргумент *служба* представляет собой имя службы, определенное в базе данных IANA (Internet Assigned Numbers Authority — Агентство по выделению имен и уникальных параметров протоколов Internet). Получить доступ к этой базе данных можно по адресу `www.iana.org/numbers.htm`. Аргумент *протокол* должен быть равен `tcp` либо `udp`. Аргумент *имя* — это домен, на который ссылается запись SRV. Аргумент *приоритет* имеет тот же смысл, что и в записи MX. Аргумент *вес* используется для распределения нагрузки между несколькими серверами, *порт* — это номер порта, на котором выполняется служба, а *сервер* — имя сервера, предоставляющего данную услугу. В ответ на запрос к записи SRV обычно возвращается запись A сервера. Имя сервера, равное ".", означает, что служба недоступна на данном хосте. Если вес равен 0, то специального распределения нагрузки не требуется.

Ниже показан пример, взятый из документов RFC2052 и RFC2782 (где определена запись SRV) и адаптированный для домена `cs.colorado.edu`:

```
_ftp._tcp       SRV  0  0  21   ftp-server.cs.colorado.edu.

; доступ к службе Finger закрыт (имя сервера -- '.')
_finger._tcp    SRV  0  0  79   .

; одна четверть соединений обслуживается старым компьютером,
; а три четверти -- новым
_ssh._tcp       SRV  0  1  22    old-slow-box.cs.colorado.edu.
                SRV  0  3  22    new-fast-box.cs.colorado.edu.

; основной сервер доступен через порт 80,
; а резервный -- через порт 8000 на новом компьютере
_http._tcp      SRV  0  0  80    www-server.cs.colorado.edu.
                SRV  10 0  8000  new-fast-box.cs.colorado.edu.

; в адресной строке можно указывать как http://www.cs.colorado.edu,
; так и http://cs.colorado.edu
_http._tcp.www  SRV  0  0  80    www-server.cs.colorado.edu.
                SRV  10 0  8000  new-fast-box.cs.colorado.edu.

; все остальные службы блокированы
*._tcp          SRV  0  0  0    .
*._udp          SRV  0  0  0    .
```

В этом примере демонстрируется использование аргументов *вес* (служба SSH) и *приоритет* (служба HTTP). Будут задействованы оба сервера SSH, причем нагрузка между ними распределяется в соответствии с производительностью серверов. Резервный сервер HTTP используется только в том случае, когда основной сервер не функционирует. Программа `finger` недоступна на данном хосте, как и все остальные службы, имена которых не упомянуты явно. Тот факт, что сведения о демоне `fingerd` отсутствуют в базе данных DNS, не означает, что он не выполняется; просто его нельзя найти средствами DNS.

Раньше в базе данных DNS существовала запись WKS (well-known services — известные службы). Вместо того чтобы переадресовывать клиента на хост, предоставляющий указанную услугу в рамках домена, она содержала список всех служб, поддерживаемых заданным хостом. Эта запись является практически бесполезной и к тому же считается рискованной с точки зрения безопасности, поэтому она не получила распространения.

Записи TXT

Запись TXT добавляет в базу данных DNS произвольный текст. Например, в нашей базе данных имеется следующая запись, идентифицирующая организацию:

```
IN          TXT  "University of CO, Boulder Campus, CS Dept"
```

Эта запись стоит непосредственно после записей SOA и NS зоны “cs.colorado.edu.”, наследуя от них поле *имя*.

Запись TXT имеет такой формат:

имя [*ttl*] IN TXT *информация* ...

Все информационные элементы должны быть заключены в кавычки. Это может быть как одна, так и несколько строк, каждая из которых взята в кавычки. Будьте предельно внимательны: пропущенные закрывающие кавычки приводят к разрушению базы данных DNS, поскольку все последующие записи вплоть до очередной кавычки загадочным образом исчезают.

Многие администраторы используют записи TXT для публикации имен компьютеров на сайтах, с которых посылаются легитимные почтовые сообщения. Другие сайты могут использовать так называемые записи SPF (Sender Policy Framework — Структура политики отправителя), с помощью которых можно идентифицировать и удалять сообщения рекламного характера (“спам”). Информация записей SPF находится в части *информация* записей TXT; для SPF нет специального типа записей DNS.

Ниже показана пара примеров записей TXT:

```
sendmail.com. IN TXT "v=spf1 ip4:209.246.26.40 ip4:63.211.143.38 ip4:
   209.246.26.36 ip4:209.246.26.12 ip4:209.246.26.18 ip4:209.246.26.10 ~all"
example.com. IN TXT "v=spf1 MX PTR -all"
```

В первой строке перечисляются IP-адреса почтовых серверов домена `sendmail.com`. Если письмо вроде бы пришло из домена `sendmail.com`, но, тем не менее, не было послано ни с одного из компьютеров, IP-адрес которого включен в список записей SPF, то это письмо удаляется. Вторая строка требует, чтобы компьютер, с которого было послано сообщение, имел такие же записи MX и PTR в DNS, гарантирующие его достоверность.

Выражение `v=spf1` определяет версию протокола SPF; в данном случае определена или реализована только версия 1. Некоторые агенты транспортировки почты (включая `sendmail`, Postfix и `exim`) поддерживают обработку SPF. С помощью записей SPF мож-

но определять еще много чего; подробное рассмотрение этих записей вы найдете в в разделе 18.11.

У записей TXT нет внутреннего порядка. Следовательно, не стоит пытаться добавить в базу данных единый информационный блок, представленный в виде нескольких записей TXT: в ответ на запрос демон `named` может возвращать их в произвольном порядке.

Записи о ресурсах IPv6

IPv6 — новая версия протокола IP. Процесс принятия спецификации длится уже более десяти лет и до сих пор не завершен[8]. Своим появлением протокол IPv6 обязан острой потребности в дополнительных IP-адресах. Однако решения этой проблемы, считавшиеся временными, — протокол CIDR, система NAT и строгий административный контроль над адресным пространством — оказались столь успешными, что массовый переход к стандарту IPv6 оказался не таким уж необходимым, как это казалось поначалу. В принятии спецификации IPv6 сейчас заинтересованы страны Азии, в которых адреса IPv4 распространяются очень слабо. Если чаша весов все-таки склонится в пользу IPv6, то это будет возможно только благодаря новому поколению мобильных телефонов, имеющих IP-адреса.

🕮 Подробнее о стандарте IPv6 рассказывалось в главе 12.

Первые предложения по поддержке IPv6 в DNS были продиктованы поддержкой коллективного владения адресами IPv6 посредством записей о ресурсах A6 и DNAME. Хотя эти типы записей и упрощали перенумерацию адресов, они все равно были столь сложными, что сейчас IETF отказалась от первоначального плана использовать более простые записи AAAA для прямых преобразований и домена ip6.arpa для обратных преобразований. В этой книге мы больше не будет рассматривать записи A6, битовые строки или записи DNAME. Если эта тема вас заинтересует, вы можете почитать предыдущие издания этой книги или документы RFC.

Прямое преобразование в стандарте IPv6

Формат записи AAAA таков:

имя [*ttl*] IN AAAA *IP-адрес*

Например:

```
anchor    IN  AAAA    3ffe:8050:201:9:a00:20ff:fe81:2b32
```

Каждый фрагмент адреса, отделенный двоеточием, состоит из четырех шестнадцатеричных цифр (ведущие нули обычно опускают). Если несколько фрагментов включают в себя одни нули, они заменяются двумя двоеточиями. В адресе может быть не более одного такого сокращения.

Обратное преобразование в стандарте IPv6

В IPv4 зоны обратного преобразования располагались в домене in-addr.arpa, а зоны прямого преобразования — в других ветвях дерева доменов (например, в доменах “com” или “edu”). В IPv6 информация об обратном преобразовании, соответствующая

8 Тони Ли (Tony Li), активный член общества IETF, как-то сказал про IPv6 такую фразу: “слишком мал, слишком скор”.

записи AAAA, задается с помощью записи PTR и хранится в домене верхнего уровня `ip6.arpa`[9].

🕮 Подробнее о записях PTR в версии IPv4 можно найти в разделе 15.7.

В "полубайтовом" формате адрес, указанный в записи AAAA, инвертируется путем разложения каждого адресного фрагмента на четыре шестнадцатеричные цифры и записи их в обратном порядке с добавлением суффикса `ip6.int`. Например, следующая запись PTR соответствует показанной выше записи AAAA для хоста `anchor`:

```
2.3.b.2.1.8.e.f.f.f.0.2.0.0.a.0.9.0.0.0.1.0.2.0.0.5.0.8.e.f.f.3.ip6.int
PTR anchor.cs.colorado.edu.
```

Было бы настоящим издевательством вводить все это самостоятельно. Естественно, в реальных зонных файлах задачу можно существенно упростить, воспользовавшись директивой `$ORIGIN`. Но после знакомства с альтернативным форматом обратных привязок даже "полубайтовый" формат начинает казаться простым и понятным.

IPv6 по-прежнему остается молодым протоколом, по крайней мере, с точки зрения развертывания. Регистраторы начинают выдавать адреса, и со временем этот процесс наладится. Некоторые из корневых серверов имен недавно (в 2004 г.) начали рекламировать адреса IPv6. Однако остается открытым вопрос о поведении. Например, если имя имеет адрес IPv4, а не IPv6, и к нему производится запрос на запись AAAA, то что в таком случае нужно сделать: сказать "нет такой записи" или вернуть запись IPv4 A?

Записи, определяющие безопасность

Записи о ресурсах, связанные с DNSSEC (DNSKEY, DS, RRSIG и NSEC), нужно рассматривать в отдельной теме. Вы прочитаете о них в разделе 15.15, в котором речь пойдет о защите DNS. Эти записи существенно отличаются тем, что обычно они генерируются программными средствами и не вводятся вручную.

Директивы в файлах зон

Закончив изучение базовых записей о ресурсах, перейдем к знакомству с директивами, которые вставляются в зонный файл для модификации последующих записей. По сути, эти директивы являются директивами синтаксического анализатора, помогающими сделать зонные файлы пригодными для чтения и удобными в поддержке. Директивы или влияют на способ, благодаря которому анализатор интерпретирует последующие записи, или расширяют сами записи DNS. После того как зонный файл будет прочитан и интерпретирован, ни одна из этих директив не будет оставаться частью данных зоны (по крайней мере, не в их исходном виде). Таких директив четыре:

```
$ORIGIN     имя_домена
$INCLUDE    имя_файла
$TTL        стандартное_время_жизни
$GENERATE   аргументы
```

Директивы *должны* начинаться в первой колонке и занимать отдельную строку. Директивы `$ORIGIN` и `$TTL` определены в документах RFC и должны быть понятны всем серверам имен; директивы `$INCLUDE` и `$GENERATE` первоначально были ориентированы на BIND, однако позже были задействованы и в некоторых других реализациях DNS.

[9] Ветка обратного преобразования IPv6 в дереве имен первоначально называлась `ip6.int`.

Когда сервер имен читает зонный файл, он добавляет стандартное имя домена ("источник") к любому имени, которое не полностью определено. Первоначально источником служит домен, указанный в соответствующей инструкции `zone` конфигурационного файла сервера имен. Эту установку можно изменить в зонном файле посредством директивы `$ORIGIN`.

Использование сокращенных имен вместо полных позволяет сэкономить много времени на вводе данных и делает зонные файлы гораздо более удобными для восприятия. Например, все записи зон обратного преобразования для сети класса B с группой подсетей могут находиться в одном зонном файле, где директивы `$ORIGIN` устанавливают контекст каждой подсети. Инструкция вида

```
$ORIGIN 243.138.128.in-addr.arpa
```

будет предшествовать записям подсети 243.

Во многих организациях в зонные файлы включаются директивы `$INCLUDE`, позволяющие разделять зонные базы данных на логические блоки или хранить ключи шифрования в отдельном файле с ограниченными правами доступа. Синтаксис директивы `$INCLUDE` таков:

```
$INCLUDE имя_файла [origin]
```

Указанный файл включается в базу данных в том месте, где стоит директива. Если будет указано *origin*, то перед содержимым читаемого файла будет стоять директива `$ORIGIN`. Если имя файла — сокращенное, оно интерпретируется по отношению к домашнему каталогу работающего сервера имен.

Директива `$TTL` задает стандартное значение для поля *ttl* последующих записей. Раньше это значение можно было определить только в записи SOA. Эта директива должна предшествовать записи SOA зоны. По умолчанию время жизни измеряется в секундах, но можно задать и другие единицы измерения: часы (`h`), минуты (`m`), дни (`d`) или недели (`w`). Например, все нижеперечисленные директивы:

```
$TTL 86400
$TTL 24h
$TTL 1d
```

устанавливают значение TTL равным одному дню.

Директива `$GENERATE`, являющаяся относительно новой конструкцией, позволяет легко создавать наборы похожих записей. В основном она применяется при работе с записями CNAME в файлах зон обратного преобразования, когда маски подсетей не проходят по границе байтов в IP-адресе (см. документ RFC2317).

Формат директивы таков:

```
$GENERATE начало-конец[/шаг] левый_арг тип правый_арг [комментарий]
```

а генерируемые строки будут иметь следующий вид:

```
левый_арг тип правый_арг
```

Поля *начало* и *конец* определяют диапазон значений счетчика цикла. Для каждого значения в диапазоне создается одна строка. Ссылка на счетчик цикла в левом и правом аргументах осуществляется посредством метасимвола `$`. Если указан *шаг*, счетчик цикла будет изменяться на указанную величину. BIND 9 поддерживает директиву `$GENERATE` для записей о типах CNAME, PTR, NS, DNAME, A и AAAA. Пример приводился ранее в этом разделе.

Связующие записи: ссылки между зонами

У каждой зоны имеется набор файлов данных, серверов имен и клиентов. Но зоны должны быть соединены друг с другом, чтобы получалась связная иерархия. К примеру, домен `cs.colorado.edu` является частью домена `colorado.edu`, и в DNS должна существовать связь между ними.

Поскольку отсылки всегда возвращаются в направлении от родительских доменов к дочерним, серверу имен не нужны сведения о доменах (точнее, зонах), расположенных выше него в иерархии DNS. В то же время серверы родительского домена должны хранить IP-адреса серверов имен всех своих поддоменов. По сути, в ответ на внешние запросы могут возвращаться отсылки только к тем серверам имен, которые известны в родительской зоне.

Говоря языком DNS, родительская зона должна содержать записи NS для каждой дочерней зоны. Но в них указываются доменные имена, а не IP-адреса, поэтому родительский сервер должен преобразовать эти имена, т.е. либо выполнить обычный DNS-запрос (если при этом не возникает циклическая зависимость), либо обратиться к собственным копиям соответствующих записей A. Реализовать это требование можно двумя способами:

- хранить все необходимые записи;
- использовать усеченные зоны.

В первом случае в базу данных родительской зоны включаются необходимые записи NS и A. Например, зонный файл домена colorado.edu может содержать следующие записи:

```
; информация о поддоменах
cs              IN  NS   ns.cs.colorado.edu.
                IN  NS   piper.cs.colorado.edu.
                IN  NS   ns.xor.com.
ee              IN  NS   ns.ee.colorado.edu.
                IN  NS   ns.cs.colorado.edu.

; связующие записи
ns.cs           IN  A    128.138.243.151
piper.cs        IN  A    128.138.204.4
ns.ee           IN  A    128.138.200.1
```

“Посторонние” записи A называются связующими, так как они в действительности не принадлежат зоне. Они лишь повторяются здесь для того, чтобы связать новый домен с деревом имен. Отсутствие или некорректность связующих записей приводит к тому, что часть адресного пространства оказывается недоступной, и пользователи, пытающиеся обратиться по этим адресам, получают сообщения об ошибке “host unknown” (неизвестный хост).

Распространенной ошибкой является включение связующих записей для тех доменных имен, которые этого не требуют. Например, в показанном выше примере адрес домена `ns.xor.com` может быть найден путем обычного DNS-запроса. Запись A будет для него поначалу просто лишней, но впоследствии окажет “медвежью” услугу, если адрес домена по какой-то причине изменится. Правило гласит, что записи A должны включаться только для хостов, которые находятся в текущем домене или одном из его поддоменов. Существующие версии BIND игнорируют ненужные связующие записи, фиксируя в журнальном файле их присутствие как ошибку.

Описанная схема представляет собой стандартный способ соединения зон между собой, но она требует, чтобы администраторы дочерних зон держали связь с администраторами родительских зон и уведомляли их об изменениях своих серверов имен. Однако из-за того, что такие зоны часто контролируются разными организациями, это становится утомительной задачей, требующей координации усилий нескольких администраторов. Как результат, использование подобной конфигурации часто приводит к хранению устаревших записей.

Второй способ обслуживания межзонных связей заключается в использовании усеченных зон. Усеченная зона, по сути, представляет собой ту же подчиненную зону, но содержит только записи NS. При автоматическом обновлении усеченной зоны не нужно поддерживать связь между родительской зоной и дочерней зоной. Серьезным недостатком является тот факт, что усеченные зоны должны быть одинаково сконфигурированы на главных и подчиненных серверах родительской зоны, что само по себе является труднодостижимой задачей. Поэтому лучше всего просто вручную держать связь с родительским доменом и проверять его конфигурацию хотя бы несколько раз в год (особенно если речь идет о локальном домене).

С помощью команды `dig` можно узнать, какие из серверов в настоящий момент известны родительскому домену. Сначала введите команду

dig *родительский_домен* **ns**

чтобы узнать серверы имен родительского домена. Выберите один из них и введите

dig @*сервер_имен.родительский_домен дочерний_домен* **ns**

чтобы получить список собственных общедоступных серверов имен. Приведем пример:

```
% dig colorado.edu ns
;;    ...
;; ANSWER SECTION:
colorado.edu.       5h9m22s IN NS  ns1.westnet.net.
colorado.edu.       5h9m22s IN NS  rs0.netsol.com.
colorado.edu.       5h9m22s IN NS  boulder.colorado.edu.
colorado.edu.       5h9m22s IN NS  cujo.colorado.edu.

% dig @boulder.colorado.edu cs.colorado.edu ns
;;; ANSWER SECTION:
cs.colorado.edu.     6H IN NS      cs.colorado.edu.
cs.colorado.edu.     6H IN NS      huizil.cs.colorado.edu.
cs.colorado.edu.     6H IN NS      xor.com.
cs.colorado.edu.     6H IN NS      pacifier.com.
```

Только четыре сервера домена `cs.colorado.edu` видны извне. Но если выполнить команду `dig` внутри факультета, будет получен другой список:

```
;; ANSWER SECTION:
cs.colorado.edu.     2H IN NS      cs.colorado.edu.
cs.colorado.edu.     2H IN NS      moet.cs.colorado.edu.
cs.colorado.edu.     2H IN NS      piper.cs.colorado.edu.
cs.colorado.edu.     2H IN NS      anchor.cs.colorado.edu.
cs.colorado.edu.     2H IN NS      vulture.cs.colorado.edu.
```

Обратите внимание, что значения времени жизни варьируются от 2 до 6 часов, в зависимости от того, откуда поступает запрос: от внешних клиентов, или от внутренних. Это объясняется тем, что для определения внутренних и внешних видов данных использовалась инструкция `view` (о ней речь пойдет в разделе 15.11). (Оба значения соответ-

ствуют короткому периоду; приемлемым вариантом будет значение, определяющее несколько дней или неделю.)

Усеченные зоны особенно удобны, когда в системе используются адреса из частного диапазона (см. документ RFC1918) и необходимо поддерживать синхронизацию с делегированными зонами. В примере (см. раздел 15.12), взятом на сайте `isc.org`, широко задействованы усеченные зоны.

Следует упомянуть о некоторых особенностях усеченных зон.

- Усеченные зоны не содержат авторитетных копий данных зоны, и усеченные серверы не должны указываться в записях NS зоны.
- Поскольку усеченные серверы не упоминаются в записях NS, они не уведомляются автоматически при изменении данных зоны. Усеченные серверы просто ожидают, пока закончится период обновления зоны, указанный в записи SOA. Если этот период будет слишком длинным, может возникнуть кратковременное напрасное делегирование (эта ситуация описывается в разделе 15.16).
- Теоретически серверу имен не нужны копии записей NS зоны, если он не может получить соответствующие записи A. Но он способен сам себя проинициализировать, используя IP-адрес главного сервера, содержащийся в его конфигурационном файле (предложение `masters` инструкции `zone`; см. раздел 15.11).
- Зачем ограничивать себя записями NS? Почему бы не создать вторичный сервер для поддомена? Этот прием тоже работает. Но если каждый сервер родительского домена одновременно является сервером дочернего домена, то отсылки на нижестоящие серверы никогда не будут генерироваться. Серверы родительского домена будут предоставлять поддомену все функции DNS. Возможно, это именно то, что нужно, а возможно — нет.

Итак, мы с вами рассмотрели много вопросов, касающихся как самой системы DNS, так и ее базы данных. В следующем разделе мы продолжим рассмотрение DNS и сосредоточимся на деталях конфигурации, специфических для реализации BIND.

15.8. Пакет BIND

BIND (Berkeley Internet Name Domain — система доменных Internet-имен реализации университета Беркли) представляет собой открытый программный пакет, предлагаемый организацией ISC (Internet Systems Consortium — консорциум разработчиков программного обеспечения для Internet). Он реализует протокол DNS и функционирует в качестве сервера имен на платформах Linux, UNIX, MacOS и Windows.

Версии пакета BIND

Имеются три основные разновидности пакета: BIND 4, BIND 8 и BIND 9. Первая из них появилась в конце 80-х гг. (ее выпуск примерно соответствует времени появления документов RFC1034 и RFC1035). Вторая была выпущена в 1997 г., а третья — в середине 2000 года. Версий 5, 6 и 7 никогда не было. Существовала красивая легенда, будто бы версия 8 содержала в себе столь значительные изменения, что авторы решили увеличить ее номер в два раза по сравнению с предыдущей[10]. Это, конечно же, не так. Просто пакет BIND 8 предназначался для 4.4BSD (Berkley Software Distribution of Unix — вер-

[10] Кто сказал, что маркетинг и проектирование не могут уживаться друг с другом?

сия системы Unix, разработанная в Калифорнийском университете), где номера версий программных продуктов были доведены до восьми (то же самое произошло с системой sendmail, которая также "перескочила" через несколько номеров).

Пакет BIND 8 содержит множество технических новинок, способствующих повышению эффективности, надежности и безопасности. В BIND 9 планка поднялась еще выше: поддерживаются многопроцессорные системы, потоковые операции, реальные технологии безопасности (шифрование с открытым ключом), стандарт IPv6, инкрементные зонные пересылки и многое другое. Новая структура базы данных (новая в BIND) — красно-черное дерево — позволяет хранить данные о зоне в памяти. Пакет BIND 9 оказался полностью переписанным и по сути реализованным заново. В нем изолированы платформо-зависимые фрагменты кода, благодаря чему стало легче переносить пакет в разные операционные системы. Внутренняя структура пакета BIND 9 существенно изменилась, но процедура конфигурирования осталась той же. В этой книге речь будет идти только о BIND 9.

Определение текущей версии

Поставщики операционных систем часто не указывают, какие версии внешних программных пакетов они включают в свои дистрибутивы. Поэтому пользователям приходится играть роль ищеек, чтобы узнать, какая версия установлена в системе. В случае пакета BIND получить интересующую информацию можно с помощью утилиты dig, входящей в состав пакета. Команда

```
dig @сервер version.bind txt chaos
```

возвращает номер версии при условии, что эта информация не была удалена из конфигурационного файла BIND. К примеру, команда выдает нужный результат для сервера isc.org:

```
% dig @ns-ext.isc.org version.bind txt chaos
version.bind.      OS CHAOS TXT   "9.4.0a0"
```

но в случае сервера cs.colorado.edu она бессильна:

```
% dig @mroe.cs.colorado.edu version.bind txt chaos
version.bind.      OS CHAOS TXT   "wouldn't you like to know..."
```

Некоторые администраторы конфигурируют пакет BIND так, чтобы номер его версии нельзя было узнать запросами подобного рода. Считается, что это повышает безопасность системы. Мы не одобряем подобную практику, хотя она действительно способна поумерить пыл некоторых юных хакеров. Подробнее о задании версии рассказывается в разделе 15.11.

Этот запрос возвращает комбинацию "0S". 0 — это цифра ноль, соответствующая времени жизни. S — это количество секунд (от англ. "second"), хотя по умолчанию время жизни измеряется в секундах, поэтому оно не печатается. Не исключено, что с первого раза эта комбинация покажется ошибочной, так как можно подумать, что речь идет об операционной системе (OS).

Комитет IETF стандартизировал данные в этой зоне класса CHAOS и обобщил их, так что другие реализации тоже могут использовать это условие. Многие сейчас так и делают, оставляя переменной имя *версия*.bind; других раздражает то, что в имени присутствует обозначение bind. Так как в IETF еще не решили, какими должны быть имена переменных, ISC реализовал все варианты в своем текущем выпуске. Например, в классе CHAOS можно запросить *имя_хоста*.bind или использовать более обобщенную запись id.*сервер*.

О системе Syslog рассказывалось в главе 10.

Обычно можно определить версию BIND, просматривая журнальные файлы в каталоге `/var/log`. Например, демон `named` при запуске регистрирует номер своей версии через систему Syslog (средство `daemon`). Поиск с помощью команды `grep` дает такие результаты:

```
Feb 23 00:25:13 senna named[433]: starting BIND 9.4.0a0 -c
   /var/named/named.ns-ext.conf
```

Если демон `named` инсталлирован, но не запускается на этапе начальной загрузки, выполните команду `named -v`, после чего он покажет номер своей версии и завершит работу.

В табл. 15.7 указано, какие версии BIND входят в состав наших демонстрационных дистрибутивов. В системах Debian и Ubuntu у вас будет пакет `bind`, инсталлирующий BIND 8, и пакет `bind9`, инсталлирующий BIND 9. В версиях, предшествующих 9.3.1 или 8.4.6, имеются проблемы с безопасностью; забудьте о версии BIND 4. Безопаснее всего использовать текущую версию.

Таблица 15.7. Версии BIND в различных дистрибутивах Linux

Система	Версия ОС	Версия BIND
ISC	—	9.4.0b
RHEL	4.3	9.2.4 через rpm
Fedora	FC5	9.3.2
SUSE	10.2	9.3.2
Debian	3.2b	8.4.6 или 9.2.4
Ubuntu	6.06	8.4.6 или 9.3.2

Многие дистрибьюторы Linux совершенствуют старые версии, повышая их безопасность.

Компоненты пакета BIND

В пакет BIND входят три компонента:

- Демон сервера имен `named`, отвечающий на запросы;
- библиотечные функции, контактирующие с серверами распределенной базы данных DNS;
- утилиты `nslookup`, `dig`, `host`, позволяющие выполнять DNS-запросы из командной строки.

Согласно терминологии, принятой в DNS, демон вроде `named` (или компьютер, на котором он работает) называется *сервером имен* (name server), а программа-клиент, которая обращается к нему, — *распознавателем* (resolver). Ниже вкратце рассматриваются функции всех этих компонентов, а реальная конфигурация пакета BIND описывается в разделе 15.11.

Демон named: сервер имен пакета BIND

Демон `named` отвечает на запросы об именах и IP-адресах компьютеров. Если демону неизвестен ответ на какой-нибудь запрос, он опрашивает другие серверы и помещает результаты их ответов в кэш. Кроме того, демон осуществляет так называемые *зонные пересылки*, копируя данные между серверами домена. (Напомним, что *зона* — это домен без поддоменов. Серверы имен работают именно с зонами, но часто говорится "домен", хотя на самом деле подразумевается "зона".)

Серверы имен работают в нескольких режимах, различия между которыми пролегают сразу в нескольких аспектах, поэтому дать их точное определение нелегко. Ситуация усложняется еще и тем, что один и тот же сервер может выполнять неодинаковые функции в разных зонах. В табл. 15.8 перечислены прилагательные, употребляемые при описании серверов имен. Определения, данные с отступом, относятся к более крупной категории.

Таблица 15.8. Классификация серверов имен

Тип сервера	Описание
Авторитетный (authoritative)	Официальный представитель зоны
главный (master)	Основное хранилище данных зоны; берет информацию из дискового файла
подчиненный (slave)	Копирует данные с главного сервера
усеченный (stub)	Напоминает подчиненный сервер, но копирует данные только о серверах имен (записи NS)
внутренний (distribution)	Сервер, доступный только в пределах домена* (также называется невидимым сервером)
Неавторитетный** (nonauthoritative)	Отвечает на запросы, пользуясь данными из кэша; не "знает", являются ли эти данные корректными
кэширующий (caching)	Кэширует данные, полученные от предыдущих запросов; обычно не имеет локальных зон
переадресующий (forwarder)	Выполняет запросы от имени многих клиентов; формирует большой кэш
Рекурсивный (recursive)	Осуществляет запросы от имени клиента до тех пор, пока не будет найден ответ
Нерекурсивный (nonrecursive)	Отсылает клиента к другому серверу, если не может получить ответ на запрос

* Внутренний сервер доступен любому, кто знает его IP-адрес.

** Строго говоря, атрибут "неавторитетный" относится к ответу на DNS-запрос, а не к самому серверу.

Серверы имен систематизируются на основании источника данных (авторитетный, кэширующий, главный, подчиненный), типа хранимых данных (усеченный), пути распространения запроса (переадресующий), типа выдаваемого ответа (рекурсивный, нерекурсивный) и, наконец, доступности сервера (внутренний). В следующих подразделах конкретизируются наиболее важные из этих различий; об остальных различиях пойдет речь в других разделах главы.

Авторитетные и кэширующие серверы

Главный, подчиненный и кэширующий серверы различаются только двумя характеристиками: откуда поступают данные и авторитетен ли сервер для домена.

В каждой зоне есть один главный сервер имен. На нем хранится официальная копия данных зоны (в файле на диске). Системный администратор модифицирует информацию, касающуюся зоны, редактируя файлы главного сервера.

Подчиненный сервер копирует свои данные с главного сервера посредством операции, называемой *зонной пересылкой* (zone transfer). В зоне должен быть как минимум один подчиненный сервер. Усеченный сервер — особый вид подчиненного сервера, загружающий с главного сервера только записи NS (описания серверов имен). О том, зачем создается такой сервер, пойдет речь в разделе 15.11. Один и тот же компьютер может быть главным сервером для одних зон и подчиненным — для других.

О зонных пересылках рассказывается в разделе 15.14.

Кэширующий сервер имен загружает адреса серверов корневого домена из конфигурационного файла и накапливает остальные данные, кэшируя ответы на выдаваемые им запросы. Собственных данных у кэширующего сервера нет, и он не является авторитетным ни для одной зоны, за исключением, пожалуй, зоны хоста `localhost`.

Гарантируется, что авторитетный ответ сервера имен является точным; неавторитетный ответ может быть устаревшим. Тем не менее, очень многие неавторитетные ответы оказываются совершенно корректными. Главные и подчиненные серверы авторитетны для своих зон, но не для кэшированной информации о других доменах. По правде говоря, даже авторитетные ответы могут быть неточными, если системный администратор изменил данные главного сервера и забыл распространить их (или не изменил порядковый номер данных).

Главный сервер зоны должен находиться на компьютере, который стабилен, имеет небольшое количество пользователей, относительно безопасен и подключен через источник бесперебойного питания. Подчиненных серверов имен должно быть минимум два, причем один из них — вне организации. Подчиненные серверы на местах должны быть включены в разные сети и в разные цепи питания. Если служба имен вдруг перестанет функционировать, пользователи не смогут нормально работать в сети.

Кэширующие серверы неавторитетны, зато позволяют сократить объем DNS-трафика во внутренней сети и уменьшить время, затрачиваемое на выполнение DNS-запросов. Желательно создавать один такой сервер в каждой подсети. В большинстве организаций DNS-запросы от настольных компьютеров обычно проходят через кэширующий сервер.

В BIND 4 и BIND 8 не рекомендовалось делать один и тот же сервер имен авторитетным для ваших собственных зон и кэширующим — для ваших пользователей. Каждый демон `named` имел единую резидентную базу данных, и когда из-за нехватки памяти данные выгружались на диск, могло произойти смешение. В BIND 9 эта проблема устранена. Тем не менее, в целях безопасности и, так сказать, “общей гигиены” DNS, желательно разделять функции передачи ваших авторитетных данных во внешний мир и функций передачи данных из внешнего мира вашим пользователям.

Рекурсивные и нерекурсивные серверы

Серверы имен бывают рекурсивными и нерекурсивными. Если у нерекурсивного сервера есть адрес, оставшийся в кэше от одного из предыдущих запросов, или он ав-

торитетен для домена, к которому относится запрашиваемое имя, то он даст соответствующий ответ. В противном случае он вернет отсылку на авторитетные серверы другого домена, которые с большей вероятностью ответят на запрос. Клиенты нерекурсивного сервера должны быть готовы принимать отсылки и обрабатывать их.

У нерекурсивных серверов обычно есть веские причины не выполнять дополнительную работу. Например, все корневые серверы и серверы доменов верхнего уровня являются нерекурсивными, поскольку им приходится обрабатывать до 10 000 запросов в секунду.

Рекурсивный сервер возвращает только реальные ответы или сообщения об ошибках. Он сам отслеживает отсылки, освобождая от этой обязанности клиента. Базовая процедура анализа запроса, по сути, остается неизмененной. По причине безопасности серверы имен организации, доступные из внешнего мира, всегда должны быть нерекурсивными.

Библиотечные функции распознавания имен не понимают отсылок. Они ожидают, что локальный сервер имен является рекурсивным.

При отслеживании отсылок возникает побочный эффект: в кэше сервера имен накапливается информация о промежуточных доменах. При работе в локальной сети от этого обычно только польза, поскольку в последующих операциях поиска, инициируемых компьютерами этой сети, можно будет пользоваться результатами предыдущих запросов. С другой стороны, сервер домена верхнего уровня (такого как “com” или “edu”) не должен хранить информацию, запрашиваемую компьютером, который находится несколькими уровнями ниже.

В первых версиях BIND для изменения рекурсивности сервера требовалось модифицировать и перекомпилировать исходный код пакета пакета. Затем появилась опция командной строки `-r`, которая теперь преобразована в параметр конфигурационного файла. Можно даже сконфигурировать сервер таким образом, что он будет рекурсивным для внутренних клиентов и нерекурсивным — для внешних.

Отсылки генерируются на иерархической основе. Если сервер, к примеру, не сможет узнать адрес хоста `lair.cs.colorado.edu`, он выдаст отсылки к серверам доменов `cs.colorado.edu`, `colorado.edu`, `edu` или корневого домена. Отсылка должна включать адреса серверов того домена, на который она указывает, поэтому выбор домена не случаен: сервер должен ссылаться на домен, серверы которого ему известны.

Как правило, возвращается отсылка на наиболее полный из известных доменов. В нашем примере в первую очередь были бы выданы адреса серверов домена `cs.colorado.edu`, при условии, что они известны. Если этот домен неизвестен, но известен домен `colorado.edu`, были бы выданы адреса его серверов имен, и т.д.

Сервер имен предварительно наполняет свой кэш содержимым файла “подсказок”, в котором перечислены серверы корневого домена. Благодаря этому всегда можно сделать отсылку, даже если она гласит: “Спроси у корневого сервера”.

Функции распознавания имен

Клиенты ищут соответствия между именами компьютеров и их IP-адресами, вызывая библиотечную функцию `gethostbyname`. В исходной версии этой функции поиск имена выполнялся в файле `/etc/hosts`. Для того чтобы такую информацию выдавала система DNS, функция должна делать запросы к модулю распознавания (распознавателю), который знает, как находить серверы имен и взаимодействовать с ними. Распознаватель обычно интегрируется в стандартные библиотеки, которые будут использоваться при компиляции приложений.

В большинстве систем функция `gethostbyname` может черпать информацию из нескольких источников: обычных файлов (например, `/etc/hosts`), DNS и даже локальной административной СУБД наподобие NIS. Файл "переключения служб" позволяет осуществлять четкий административный контроль над тем, какие источники опрашиваются и в каком порядке. Подробнее об этом рассказывается в разделах 15.17 и 17.15. А в разделе 15.17 вы найдете подразделы, в которых будут описаны подробные сведения о ПО сервера имен `named`.

Приложения, использующие сеть, обычно работают с усеченной библиотекой распознавателя, которая посылает DNS-запросы локальному кэшированному серверу имен. В протоколе IPv6 с этим все обстоит сложно, а вот BIND 9 предлагает "упрощенную библиотеку распознавателя" и демона распознавателя — `lwresd` — для сайтов, которым для общения не нужен протокол IPv6. Употребление термина "упрощенный" в контексте DNS может показаться неуместным, однако в данном случае он указывает на протокол, используемый между библиотекой распознавателя и демоном распознавателя. Если в конфигурационный файл сервера имен `named` включить инструкцию `lwres`, то и сам сервер имен будет функционировать в качестве упрощенного распознавателя. Пока что демон распознавателя не использует упомянутый выше файл переключения служб: для распознавания имен он работает только с DNS.

Выполнение запросов из командной строки

В пакет BIND входят утилиты `dig`, `host` и `nslookup`, позволяющие выполнять DNS-запросы в режиме командной строки. Они используются при отладке и для извлечения информации из базы данных DNS. Все эти утилиты обладают сходными функциональными возможностями, но реализованы по-разному. Подробнее о них рассказывается в разделе 15.16.

15.9. Проектирование среды DNS

Проект устойчивой и эффективной системы DNS в вашей конкретной среде зависит от многих факторов, среди которых можно выделить следующие:

- размер вашей организации;
- используются ли в вашей локальной сети приватные IP-адреса, определенные в документе RFC1918;
- используете ли вы DHCP;
- используете ли вы службу каталогов Microsoft Windows Active Directory;
- что применяется в вашей внутренней сети: маршрутизация или коммутация;
- где расположен брандмауэр по отношению к вашим серверам DNS.

Данную проблему удобно разделить на три части:

- управление иерархией пространства имен: поддомены, уровни и т.п.;
- передача авторитетных данных о вашем сайте внешнему миру;
- предоставление пользователям возможности производить поиск имен.

Управление пространством имен

Если размеры вашего сайта небольшие, и он является независимым, то поддомены лучше не использовать, если для этого нет каких-то нетехнических причин. С другой стороны, в средней организации с несколькими независимыми группами системных администраторов пододмены могут упростить координацию сайта. (Чаще всего встречаются поддомены, разделенные по географическим или ведомственным признакам.) В крупных организациях нет возможности использовать уникальные имена в пределах всего сайта, поэтому использование поддоменов, возможно даже многоуровневых, неизбежно.

В процессе построения поддоменов обязательным условием является координированная работа системных администраторов, отвечающих за родительский домен, и администраторов, отвечающих за поддомен. На момент делегирования и настройки поддомена нужно четко определиться с тем, к кому вы будете обращаться, когда захотите добавить, изменить или удалить серверы. Кроме того, ваш брандмауэр не должен блокировать доступ к серверам поддомена, если вы хотите, чтобы к поддомену был открыт доступ из-за пределов вашей организации.

Если вы используете поддомены для управления вашим пространством имен, запускайте раз в неделю инструмент `doc` (domain obscenity control — контроль за "чистотой" использования домена) из `cron`, чтобы следить за синхронизацией ваших делегирований и не допускать напрасных делегирований. Описание этого инструмента, а также других средств, помогающих обслуживать DNS, вы найдете в разделе 15.16.

Авторитетные серверы

Спецификации DNS требуют, чтобы на каждый домен приходилось как минимум по два авторитетных сервера. Авторитетными являются главный и подчиненные серверы; кэширующие и усеченные серверы не являются таковыми. В идеале, сайт должен иметь несколько авторитетных серверов, каждый из которых должен обслуживать отдельную сеть и иметь отдельную цепь питания. Многие сайты используют авторитетные серверы, размещенные вне организации — например, у своего провайдера услуг Internet. Если ваш провайдер не предлагает такую услугу, вы можете приобрести ее у провайдера услуг DNS или у местной фирмы (в идеале, не у конкурента) или учебного заведения.

Несколько лет назад Microsoft нарушила правило раздельных сетей. Три ее авторитетных сервера находились в одной и той же подсети, и когда произошел сбой в маршрутизаторе, соединяющем данную подсеть с сетью Internet, все серверы оказались недосягаемыми. Несколько часов спустя, когда истек срок нахождения записей в кэше, `microsoft.com` и все остальные его домены прекратили свою работу в Internet. Количество запросов к именам, имеющим отношение к Microsoft, на корневых серверах возросло до 25% от общей нагрузки (10 000 запросов в секунду), причем исходная величина составляла 0,000001%. Проблема просуществовала пару дней. Когда все утряслось, в Microsoft починили маршрутизатор и поменяли провайдера DNS!

Авторитетные серверы синхронизируют свои данные благодаря использованию зонных пересылок. Чтобы идентифицировать и управлять зонными пересылками из главного сервера на подчиненные серверы, следует использовать ключи TSIG. Сведения о конфигурации TSIG вы найдете в разделе 15.15.

Иногда бывает необходимо сделать так, чтобы ответы на запросы, генерируемые вашими авторитетными серверами, в той или иной мере зависели от того, кто генерирует запросы. Для запроса, поступившего из-за пределов вашей сети, может быть сгенериро-

ван один ответ, в то время как для такого же запроса из вашей сети может быть сгенерирован отличный (более полный) ответ. Такая конфигурация называется "расщеплением DNS" и реализуется на уровне зоны, а не на уровне сервера.

Каждая версия зоны называется "представлением" (view); она указывается после инструкции `view`, с помощью которой представление конфигурируется в конфигурационном файле BIND. Внешние пользователи будут видеть одно представление данных, а внутренние пользователи — другое. Эта особенность обычно используется для того, чтобы скрыть существование внутренних компьютеров от пытливых глаз, и чтобы гарантировать, что компьютеры, использующие IP-адреса из частного диапазона (RFC1918), не смогут передать информацию о них в Internet. Отладка представлений — дело нелегкое, хотя в этом отношении могут помочь обширные возможности BIND для регистрации и умелое использование команды `dig`; некоторые советы по этому вопросу вы найдете в разделе 15.16.

Кэширующие серверы

Рекурсивные кэширующие серверы отвечают на запросы локальных пользователей о сайтах в сети Internet. Каждый компьютер в вашей организации должен иметь доступ на чтение в локальном кэширующем сервере, желательно в таком сервере, который находится в той же подсети.

Организациям следует прибегнуть к внедрению иерархической системы, в которой один или несколько компьютеров выступают в роли "ретранслятора", через который кэширующие серверы локальных подсетей передают свои запросы. Таким образом, ретрансляторы организовывают кэш, являющийся общим для множества подсетей.

В зависимости от размеров сайта ретрансляторы могут быть или независимыми, или систематизированными в иерархии. Например, серверы подсети могут перенаправлять данные на сервер департамента предприятия, который, в свою очередь, перенаправляет данные дальше — на шлюз сайта. Ретранслятор, обслуживающий весь сайт, кэширует все имена, запрашиваемые пользователями в рамках сайта. Такая конфигурация позволяет сузить внешнюю полосу пропускания, выделяемую для службы имен, и позволяет всем локальным компьютерам совместно использовать один большой кэш. Ретрансляторы будут рассматриваться в разделе 15.11.

Если произойдет сбой в кэширующем сервере, то сеть перестанет обслуживать всех пользователей, которые являлись клиентами этого серевера. (И начнет трезвонить ваш телефон.) Если произойдет такая ситуация, воспользуйтесь сценарием, который перезапускает кэширующие серверы имен в течение нескольких секунд, прошедших с момента сбоя. Ниже пресдавлен пример сценария, выполняющегося на компьютере, на котором запущено два экземпляра сервера имен `named`. Данный сценарий принимает один аргумент, идентифицирующий журнальный файл и конфигурационный файл сервера имен:

```
#!/bin/sh
PATH=/usr/local/sbin:/usr/sbin:/sbin:$PATH
export PATH
trap " " 1
while :; do
  named -f -c /var/named/named.$1.conf >> /var/log/named.$1 2>&1 < /dev/null
  logger "named ($1) restart"
  sleep 15
done
exit
```

Этот сценарий отсчитывает 15 секунд (это произвольное значение), по истечении которых перезапускается сервер `named` и выполняется директива `logger`, регистрирующая запись в системном журнале всякий раз, когда требуется перезапустить сервер.

Защита

Вопросам защиты DNS посвящен целый раздел 15.15. Здесь мы не будем повторяться, разве что напомним, что если вы будете использовать брандмауэр, то ваша система DNS не должна генерировать запросы, ответы на которые будет блокировать брандмауэр. Это означает, что ваши администраторы DNS должны постоянно поддерживать связь с вашими администраторами системы и защиты. По умолчанию DNS использует для запросов UDP со случайными непривелегированными исходными портами (>1023); ответы представляют собой пакеты UDP, направляемые на те же порты.

Подводя итоги

На рис. 15.2 показан проект, рекомендованный в предыдущих разделах. На нем изображена двухуровневая иерархия переадресации, которая непригодна для использования в малых организациях. Вам нужно будет так сбалансировать работу серверов, обрабатывающих исходящие запросы, и серверов, обрабатывающих входящие запросы, чтобы ни один из серверов не оказался перегруженным.

Также настоятельно рекомендуется использовать подчиненный сервер, находящийся за пределами организации. Организации, имеющие большое число офисов, могут выделить один сервер в качестве главного; этот сервер окажется за пределами организации по отношению ко всем компьютерам данной организации.

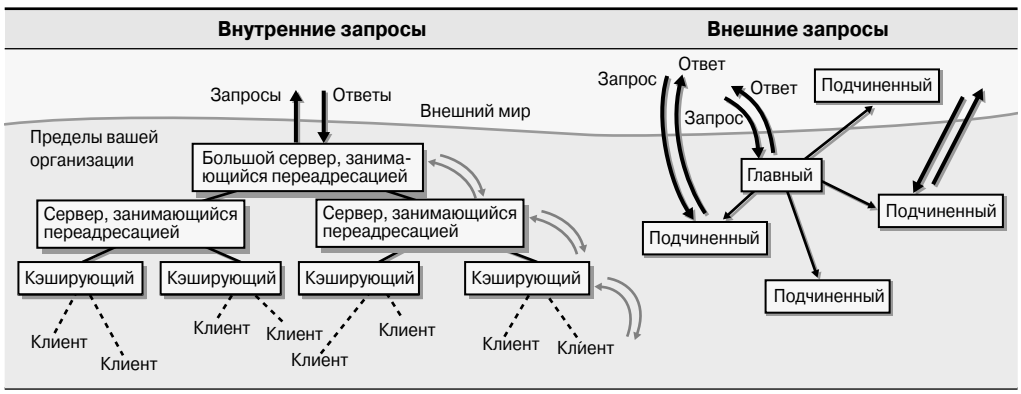


Рис. 15.2. Архитектура сервера DNS

Задачи, выполняемые с помощью DNS/BIND

В табл. 15.9 подводится итог тому, какие задачи выполняются, для кого и как часто, если речь идет об использовании BIND и DNS. Если элемент в столбце “Как часто” содержит слово “распространить”, то это означает, что указанное действие необходимо выполнить один раз для каждой подсети или платформы, а затем скопировать результат на другие компьютеры с помощью такой программы, как `rdist` или `rsync`.

О распределении файлов по сети можно прочитать в главе 17.

Таблица 15.9. Задачи, связанные с инсталляцией и сопровождением пакета BIND

Задача	Субъект	Как часто
Получение доменного имени	Организация	Один раз
Выбор серверов имен	Организация	Один раз или больше
Получение дистрибутива BIND	Организация	Один раз, но следить за обновлениями
Конфигурирование распознавателя	Клиент	Для каждой подсети и распространить
Конфигурирование файла "переключения служб"	Клиент	Для каждой платформы и распространить
Запуск демона `named` на этапе начальной загрузки	Сервер	Для каждого сервера имен
Настройка конфигурационного файла демона `named`	Сервер	Для каждого типа сервера
Конфигурирование файла "подсказок"	Сервер	Один раз* и распространить на серверы имен
Конфигурирование файлов зон	Главные серверы	Один раз
Обновление файлов зон	Главные серверы	По мере необходимости
Просмотр журнальных файлов	Регистрационный сервер	Хотя бы раз в неделю
Обучение пользователей	Все хосты	Постоянно

* Нужно выполнять повторно, если меняются корневые серверы.

15.10. Работа с клиентом BIND

Поскольку каждый компьютер в сети должен быть клиентом BIND, начнем с рассмотрения задач, стоящих перед клиентами.

Конфигурирование распознавателя

На каждом Linux-компьютере, подключенном к сети, должен находиться файл `/etc/resolv.conf` со списком DNS-серверов, которым можно посылать запросы. Если хост получает свой IP-адрес и сетевые параметры от DHCP-сервера, этот файл заполняется автоматически. В противном случае его нужно редактировать вручную. Формат записей файла таков:

```
search имя_домена ...
nameserver IP-адрес
```

Может быть указано от одного до трех серверов имен. Приведем пример:

```
search cs.colorado.edu colorado.edu ee.colorado.edu
nameserver 128.138.243.151      ; ns
nameserver 128.138.204.4        ; piper
nameserver 128.138.240.1        ; anchor
```

Для файла `resolv.conf` никогда не вводилось понятие комментариев. Они поддерживаются лишь в том смысле, что любой нераспознанный элемент файла игнорируется. Комментарии вполне безопасны в конце директивы `nameserver`, поскольку анализа-

тор файла ищет лишь IP-адрес, игнорируя остальную часть строки. А вот в директиве `search` их лучше не ставить, так как она может содержать несколько аргументов.

В строке `search` приведен список доменов, которые следует опрашивать, если имя компьютера не полностью определено.

Например, когда пользователь вводит команду `ssh foo`, распознаватель дополняет имя первым доменом в списке (в случае приведенного выше файла `resolv.conf` — cs.colorado.edu), после чего ищет хост `foo.cs.colorado.edu`. Если это имя не найдено, делается попытка найти хост `foo.colorado.edu`, затем `foo.ee.colorado.edu`. Количество доменов, которые можно указать в директиве `search`, зависит от распознавателя; как правило, это шесть-восемь доменов.

Серверы имен, указанные в файле `resolv.conf`, должны быть рекурсивными, поскольку распознаватель не понимает отсылок. Серверы имен опрашиваются в том порядке, в котором заданы директивы `nameserver`. Если первый сервер отвечает на запрос, другие серверы игнорируются. В случае превышения тайм-аута запрос переадресуется следующему серверу. Все серверы опрашиваются по очереди, до четырех раз каждый. С каждой неудачей тайм-аут увеличивается.

Большинство распознавателей допускает наличие максимум трех серверов имен. Все остальные просто игнорируются. Если хост сам является сервером имен, он должен быть указан первым в собственном файле `resolv.conf`. Если не будет указано ни одного сервера имен, значит, речь идет о локальном хосте.

В первых версиях пакета BIND вместо директивы `search` в файле `resolv.conf` использовалась директива `domain`. Она задавала единый домен, который следовало добавлять к неполным именам. Мы рекомендуем заменить директивы `domain` директивами `search`. Они являются взаимоисключающими, поэтому только одна из них должна присутствовать. Если имеется старый распознаватель, а в файле `resolv.conf` присутствуют обе директивы, эффект будет иметь та из них, которая указана последней.

Тестирование распознавателя

Многие дистрибутивы Linux будут пытаться искать имена хостов в DNS по умолчанию, если в файле `/etc/resolv.conf` будет находиться строка `nameserver`. Порядок различных источников информации об именах хостов, включая DNS, задается в файле `/etc/nsswitch.conf`; см. раздел 15.17.

После конфигурирования файла `/etc/resolv.conf` (предполагается, что соединение с локальной сетью установлено и функционирует корректно) пользователь сможет ссылаться на другие компьютеры по именам, а не IP-адресам. Если при обращении к локальному компьютеру команда “зависнет”, попробуйте найти компьютер по IP-адресу. Успешное завершение такого поиска означает, что в конфигурации DNS есть ошибка. Убедитесь в том, что IP-адреса серверов имен в файле `/etc/resolv.conf` правильны, а серверам разрешено обрабатывать запросы из вашей сети (см. описание параметра `allow-query` в разделе 15.11). На эти вопросы можно ответить с помощью команды `dig`, выполненной на работающем компьютере.

Влияние на остальную часть системы

При переходе от статических таблиц имен к DNS могут возникнуть определенные зависимости между процедурой начальной загрузки и конфигурацией системы, от которых нужно избавиться.

Ссылки на имена компьютеров в сценариях запуска системы могут оказаться неразрешимыми, если они встретятся до того, как загрузится сетевая подсистема. Команды сценариев будут безуспешно пытаться связаться с DNS, многократно повторяя попытки на множестве серверов и при каждой попытке увеличивая интервал тайм-аута. Несколько минут спустя команда, которой требуется распознать имя компьютера, наконец, выдаст ошибку.

Проблему можно решить, используя на ранних стадиях начальной загрузки только явные IP-адреса. Или же, поскольку Linux поддерживает одновременное использование DNS и файла `/etc/hosts` (благодаря файлу "переключения служб"), можно инсталлировать файл `hosts`, содержащий адреса серверов, которые необходимы на этапе начальной загрузки. При этом файл должен опрашиваться раньше, чем DNS, чтобы не пришлось ждать истечения тайм-аута.

15.11. Конфигурирование сервера BIND

В этом разделе предполагается, что глобальные административные задачи уже решены, т.е. вы получили имя домена (возможно, поддомена), связались с администратором DNS-сервера родительского домена и получили полномочия по обратному преобразованию адресов в домене in-addr.arpa. Был выбран главный сервер имен и несколько подчиненных, а также инсталлирован пакет BIND.

Аппаратные требования

Пакет BIND — настоящий пожиратель ресурсов. Его база данных хранится в памяти, поэтому одновременно с увеличением кэша растет число страниц памяти, занимаемых демоном `named`. Новые компоненты BIND 9, особенно DNSSEC, столь же интенсивно используют центральный процессор. Для снижения нагрузки пакет BIND 9 был сделан многопотоковым и теперь эффективно работает в многопроцессорных системах. Имеются также конфигурационные параметры, посредством которых можно контролировать использование системных ресурсов демоном `named`.

Лучший способ определить, достаточно ли памяти на компьютере, где установлен сервер имен, — дать серверу поработать некоторое время, а затем проверить размер процесса `named`. Требуется одна-две недели, чтобы этот размер стабилизировался и старые записи в кэше устаревали с той же скоростью, с какой появляются новые.

Если для сервера имен выделен отдельный компьютер, рекомендуется устанавливать в два раза больше памяти, чем занимает процесс `named` после двух недель работы. Статистику использования памяти можно узнать с помощью команд `top` и `vmstat` (подробнее об этом рассказывается в разделе 25.3).

Конфигурационные файлы

Полная конфигурация демона `named` включает его конфигурационный файл, файл "подсказок" корневого сервера имен и, в случае главного сервера имен, файлы данных зоны, содержащие адресные привязки для каждого хоста. Конфигурационный файл имеет специфический формат; остальные файлы представляют собой коллекции отдельных DNS-записей, отформатированных в соответствии со спецификацией DNS. Ниже пойдет речь о содержимом конфигурационного файла, тогда как о формате DNS-записей рассказывалось в разделе 15.7.

В конфигурационном файле демона named — named.conf — задается роль (главный, подчиненный или усеченный) сервера и определяется способ, которым сервер должен получать копию данных для каждой зоны, обслуживаемой сервером. Здесь же приводятся всевозможные параметры — как глобальные, связанные с работой самого демона, так и локальные, применяемые только к трафику DNS.

Конфигурационный файл состоит из набора инструкций, каждая из которых оканчивается точкой с запятой. Лексемы разделяются пробельными символами, к которым относится и символ новой строки. Иногда для группировки лексем применяются фигурные скобки. Формат файла довольно "хрупкий": достаточно одной пропущенной точки с запятой, чтобы все перестало работать.

К счастью, в BIND 9 имеются удобные средства проверки синтаксиса конфигурационного файла (named-checkconf) и зонных файлов (named-checkzone). Эти утилиты ищут не только синтаксические ошибки, но и очевидные пропуски. Например, утилита named-checkzone выдаст предупреждение, если в файл не включена директива $TTL. Но на каждую бочку меда всегда найдется ложка дегтя. Например, не выявляются пропущенные связующие записи (описаны в разделе 15.7), хотя их отсутствие приводит к повышению нагрузки на корневые серверы и серверы организационных доменов.

Комментарии допускаются везде, где могут стоять пробелы. Поддерживаются комментарии в стиле языков C, C++ и командного интерпретатора:

```
/* Это комментарий, который может занимать несколько строк. */
// Весь текст до конца строки является комментарием.
# Весь текст до конца строки является комментарием.
```

Каждая инструкция начинается с ключевого слова, определяющего ее тип. Может присутствовать несколько инструкций одного типа, за исключением options и logging. Отдельные инструкции, а также их части могут отсутствовать; в этом случае будут приняты установки по умолчанию. Список поддерживаемых инструкций приведен в табл. 15.10.

Таблица 15.10. Инструкции, используемые в файле named.conf

Инструкция	Назначение
include	Подключает внешний файл (например, файл с ключами шифрования, доступный только демону named)
options	Задает глобальные параметры конфигурации сервера имен, а также установки по умолчанию
acl	Формирует списки управления доступом
key	Определяет параметры аутентификации
trusted-keys	Задает заранее установленные ключи шифрования
server	Задает параметры сервера
masters	Определяет список главных серверов для включения в усеченные и подчиненные зоны
logging	Устанавливает категории журнальных сообщений и каналы их распространения
zone	Определяет зону, для которой демон является авторитетным
controls	Определяет, как утилита ndc будет управлять сервером имен
view	Определяет представление пространства имен
lwres	Конфигурирует сервер имен в качестве упрощенного распознавателя

Прежде чем приступать к описанию этих инструкций, необходимо рассказать о специальной структуре данных, используемой во многих инструкциях: *список соответствия адресов* (address match list). Этот список является обобщением понятия IP-адреса и может включать:

- IP-адрес, или v4, или v6 (например, 199.165.145.4);
- адрес сети с маской CIDR[11] (например, 199.165/16);
- имя ранее определенного списка управления доступом (задается с помощью инструкции `acl`);
- ключ аутентификации;
- операцию отрицания `!`.

Списки соответствия адресов используются в качестве аргументов инструкций и опций. Приведем ряд примеров:

```
{ ! 1.2.3.13; 1.2.3/24; };
{ 128.138/16; 198.11.16/24; 204.228.69/24; 127.0.0.1; };
```

В первом из этих списков из числа адресов исключается хост 1.2.3.13, но разрешаются другие адреса в сети 1.2.3/24. Во втором списке указаны сети, закрепленные за университетом штата Колорадо. Фигурные скобки и завершающая точка с запятой не являются частью списка: они включены для иллюстрации; а вообще-то они являются частью инструкций, частью которых являются списки соответствия адресов.

Когда IP-адрес или адрес сети проверяется на соответствие списку, содержимое списка просматривается по очереди до тех пор, пока не будет найдено совпадение. Порядок элементов списка важен, так как ищется первое совпадение. Например, если в первом из показанных выше списков поменять элементы местами, результат не будет достигнут, поскольку адрес 1.2.3.13 удовлетворит первому условию (сетевому адресу 1.2.3/24) и второе условие никогда не будет проверяться.

Инструкция include

Когда конфигурационный файл становится слишком большим, можно разбить его на части, поместив их в отдельные файлы. Дополнительные компоненты подключаются к файлу `named.conf` посредством инструкции `include`:

```
include "путь";
```

Если указан относительный путь, то он добавляется к имени каталога, заданному в параметре `directory` (описан ниже). Очень часто инструкцию `include` применяют для подключения файлов, содержащих ключи шифрования. Эти данные должны быть доступны только демону `named`. Чтобы не запрещать доступ ко всему файлу `named.conf`, ключи хранят в отдельных файлах, которые может читать лишь демон `named`.

Инструкция options

Инструкция `options` задает глобальные параметры конфигурации, часть из которых впоследствии может быть переопределена для конкретных зон или серверов. Общий формат инструкции таков:

[11] Сетевые маски CIDR рассматривались в разделе 12.4.

```
options {
   параметр;
   параметр;
   ...
};
```

Если в файле named.conf нет инструкции options, принимаются установки по умолчанию.

В BIND 8 существует около 50 параметров, а в BIND 9 их более 100. Полный их список можно найти в документации[12]. Ниже будут описаны лишь те параметры, которые рекомендуется задавать (мы учли мнения разработчиков пакета). Значения по умолчанию указаны в квадратных скобках рядом с каждым параметром. Для большинства организаций подойдут значения по умолчанию. Параметры перечисляются без какого-либо заданного порядка.

```
directory "путь";                    [каталог запуска сервера]
```

Этот параметр задает каталог, в который должен перейти демон named. Когда в конфигурационном файле встречаются неполные имена файлов, к ним добавляется указанный путь (сам он должен быть абсолютным). В этот каталог помещаются все выходные файлы (отладочные, статистические данные и т.д.).

Рекомендуется хранить все конфигурационные файлы пакета BIND (кроме named.conf и resolv.conf) в подкаталоге каталога /var (или любого другого каталога, где содержатся конфигурационные файлы других программ). Мы предпочитаем /var/named.

```
version "строка";                    [настоящий номер версии сервера]
hostname "строка";                   [настоящее имя сервера]
server-id "строка";                  [пусто]
```

Строка version идентифицирует версию ПО сервера имен, функционирующего на сервере. Строка hostname идентифицирует сам сервер, как и строка server-id. Эти параметры позволят вам забыть об истинных значениях. Каждый из них помещает данные в TXT-записи класса CHAOS, где любопытные наблюдатели смогут поискать их.

Имеется два принципиальных подхода к вопросу о том, нужно ли скрывать номер версии. Некоторые администраторы полагают, что серверы будут подвержены большей опасности, если хакеры узнают, какая версия пакета BIND используется. Другие считают сокрытие версии избыточной процедурой, аргументируя свою позицию тем, что хакеры в любом случае будут пытаться "взломать" систему, а большинство недавно обнаруженных ошибок присутствует во всех версиях пакета.

Мы рекомендуем не сбрасывать строку версии. Для администратора очень удобно иметь возможность запрашивать сервер имен, чтобы выяснить текущую версию пакета. Это позволяет определить, включена ли в системный дистрибутив последняя версия и все ли серверы используют одну и ту же версию.

Параметры hostname и server-id были добавлены недавно; это было связано с использованием альтернативной маршрутизации для дублирования экземпляров корневого сервера и сервера gTLD. Например, в мире насчитывается 20 экземпляров f.root-servers.net, и вам, как пользователю или системному администратору, пытающемуся произвести отладку, совершенно непонятно, какой из этих 20 серверов не отвечает на запросы (если предположить, что возникла такая проблема). Если же для обращения использовать *имя_хоста*.bind в классе CHAOS, вы могли бы узнать, какой экземпляр

[12] Файл *doc/misc/options* содержит синтаксис языка конфигурирования и включает все параметры.

корневого сервера F отвечал на запросы (попутно вы бы узнали, что схема маршрутизации не изменилась). `server-id` — это "политически" правильное имя для компьютера *имя_хоста*`.bind`, так как BIND работает не на всех серверах.

```
notify yes | no;                    [yes]
also-notify адреса_серверов;        [пусто]
```

Если параметр `notify` равен `yes`, а демон `named` является главным сервером имен одной или нескольких зон, то демон будет автоматически уведомлять подчиненные серверы этих зон при изменении соответствующих баз данных. В ответ на полученное уведомление подчиненные серверы связываются с главным сервером и обновляют свои копии данных. Параметр `notify` может устанавливаться как на глобальном уровне, так и на уровне отдельных зон. Это позволяет быстрее обновлять зонные файлы при внесении изменений в базы данных DNS.

Демон `named` выясняет, какие компьютеры являются подчиненными серверами зоны, просматривая записи NS этой зоны. При наличии параметра `also-notify` будут также уведомляться дополнительные серверы, которые не зарегистрированы посредством записей NS. Подобный прием используется, если в организации имеются внутренние серверы. В списке `also-notify` не нужно указывать усеченные серверы: их интересуют только записи NS, поэтому они могут участвовать в обычном цикле обновления. В списке должны присутствовать только IP-адреса. Уведомления желательно также отключать для зоны обратного преобразования хоста `localhost`.

```
recursion yes | no;                                  [yes]
allow-recursion { список_соответствия_адресов };     [все хосты]
```

Параметр `recursion` указывает на то, является ли демон `named` рекурсивным сервером (см. раздел 15.8). Сервер имен делают нерекурсивным лишь в очень редких случаях. Но можно, к примеру, разрешить рекурсию для внутренних клиентов демона и отключить для внешних запросов.

Параметр `allow-recursion` позволяет точнее управлять рекурсией. В нем задается список хостов и сетей, от имени которых разрешается выполнять рекурсивные запросы.

```
transfer-format one-ansfer | many-answers;          [many-answers]
```

Этот параметр влияет на способ передачи записей базы данных DNS от главных серверов к подчиненным. Раньше пересылка осуществлялась по одной записи за раз, что было очень медленно и неэффективно. Атрибут `many-answers`, позволяющий объединять несколько записей в один пакет, появился в BIND 8.1. В BIND 9 он уже установлен по умолчанию. Задавать атрибут `many-answers` следует только в том случае, когда на всех серверах, между которыми происходит обмен данными зон, инсталлирован пакет BIND версии не ниже 8.1, поскольку серверы BIND 4 не понимают этот атрибут. При работе в смешанной среде глобальные установки можно переопределять для отдельных серверов (в инструкциях `server`).

```
transfers-in число;                 [10]
transfers-out число;                [10 (только в V9)]
transfers-per-ns число;             [2]
transfer-source IP-адрес;           [зависит от системы]
```

Эти параметры, управляющие зонными пересылками, требуется настраивать в крупных серверах, управляющих очень большими зонами (такой как "com", размер базы данных которой в настоящее время превышает 2 Гбайт) или обслуживающих тысячи зон.

Параметры transfers-in и transfers-out ограничивают число входящих и исходящих зонных пересылок, которые могут происходить одновременно[13]. Параметр transfers-per-ns задает предельное количество входящих зонных пересылок, одновременно поступающих от одного удаленного сервера. В крупных серверах может потребоваться увеличить значения transfers-in и transfers-out, но будьте осторожны, так как можно превысить допустимое число файловых дескрипторов для процесса named. Параметр transfers-per-ns обычно менять не требуется. Его можно увеличить только в том случае, когда все удаленные главные серверы готовы обрабатывать более двух одновременных зонных пересылок. Если уж менять его, то лучше делать это с помощью предложения transfers инструкции server.

Параметр transfer-source позволяет указать IP-адрес сетевого интерфейса, используемого для приема входящих пересылок. Этот адрес должен соответствовать параметру allow-transfer главного сервера.

При изменении любого из перечисленных параметров нужно внимательно следить за тем, чтобы не ухудшилась производительность системы. В этом случае на помощь приходят журнальные файлы.

```
files число;          [unlimited]
datasize число        [Значение по умолчанию в ОС, не зависит от системы]
```

Параметр files задает максимальное число файлов, которые могут быть одновременно открыты на сервере. Значение по умолчанию, unlimited, означает, что необходимо максимально приблизиться к предельному числу открытых файлов, которое поддерживается ядром (для изменения этого предела предназначен файл /proc/sys/fs/file-max; см. раздел 28.4).

Параметр database задает максимальный объем памяти для данных, который разрешает использовать сервер. Единицы измеряются в байтах, хотя named воспринимает G как гигабайты, M — как мегабайты и т.д. Этот параметр не следует использовать для управления размером кэша; это жесткий лимит — он зависит от месторасположения в коде и может вызвать сбой на сервере. Обычно он используется для увеличения лимита максимального объема памяти, который по умолчанию выделяется для процесса в операционной системе. Для управления размером кэша следует использовать параметры max-cache-size и recursive-clients.

```
max-cache-size число;         [неограниченное]
recursive-clients число;      [1000]
```

Параметр max-cache-size ограничивает объем памяти, который сервер может задействовать для кэширования ответов на запросы. Если размер кэша становится слишком большим, записи будут удаляться из кэша еще до того, как истечет их время жизни, чтобы память кэша использовалась в заданных пределах.

Параметр recursive-clients ограничивает количество рекурсивных запросов, которые сервер может обрабатывать одновременно. Для каждого рекурсивного запроса отводится порция памяти на сервере; для серверов с ограниченным объемом памяти этот лимит можно сделать меньше, чем принятый по умолчанию.

```
query-source address IP-адрес port IP-порт;          [случайные значения]
listen-on port IP-порт список_соответствия_адресов; [53, все интерфейсы]
```

Параметр query-source задает интерфейс и порт, через которые демон named будет опрашивать другие серверы имен. Параметр listen-on определяет сетевые интерфейсы

[13] В исходном коде пакета BIND для параметра *transfers-in* задан жесткий лимит: 20.

и порты, по которым демон named ожидает поступления запросов. Если ни порты, ни IP-адреса не указаны, демон поступает так: прослушивает порт 53 по всем интерфейсам и посылает запросы через произвольный интерфейс, используя непривилегированный UDP-порт, выбранный случайным образом.

Если имеется брандмауэр, настройте должным образом параметр query-source; это позволит распознавать внешние DNS-запросы. Обычно выбирается порт 53, благодаря чему брандмауэр распознает исходящий DNS-трафик как поступающий от одного из надежных серверов имен. Однако это условие усложняет фильтру пакетов различать входящий и исходящий трафик DNS. Вместо порта 53 можно использовать другой порт, лишь бы только брандмауэр мог различать эти порты.

Не задавайте нулевой порт: это неправильная установка. Демон named будет регистрировать сообщения об ошибке и не отвечать на такие запросы.

```
avoid-v4-udp-ports [список_портов];              [пусто]
avoid-v6-udp-ports [список_портов];              [пусто]
```

Если не использовать параметр query-source, BIND 9 будет использовать любой исходящий интерфейс и выборочным образом назначать для запроса исходный порт из диапазона непривилегированных портов (>1023). Если ваш брандмауэр блокирует некоторые порты из этого диапазона, у вас сразу же возникнет проблема. Так, например, некоторые брандмауэры неправильно воспринимают порт 1024 — они считают, что он попадает в диапазон привилегированных портов; или же, они правильно поступают, блокируя доступ к порту 2049 (SunRC) из внешнего мира. Когда сервер имен посылает запрос и использует один из заблокированных портов как его исходный порт, брандмауэр блокирует ответ и сервер имен, выждав некоторую паузу, вновь посылает запрос. В этом нет фатальной ошибки, однако пользователь должен будет знать, что в системе что-то происходит не так.

Чтобы эта проблема не возникла, воспользуйтесь параметром avoid-v4-udp-ports, чтобы BIND не работала с заблокированными портами. Любые порты UDP с высокими номерами, блокируемые вашим брандмауэром, должны стоять первыми в списке[14]. Если вы будете обновлять брандмауэр в ответ на какую-нибудь угрожающую атаку, не забудьте обновить также и список портов.

```
forwarders { IP_адрес; IP_адрес; ... };     [пустой список]
forward only | first;                       [first]
```

Вместо того чтобы каждый сервер имен выполнял собственные внешние запросы, один или несколько серверов можно сделать *переадресующими* (forwarder). В такой конфигурации обычный сервер просматривает в своем кэше записи, для которых он авторитетен, и, если не находит ответ, посылает запрос переадресующему серверу. Последний создает кэш, которым пользуются все остальные серверы. В конфигурационном файле переадресующего сервера нет ничего такого, на основании которого можно было бы сказать: “Это переадресующий сервер”.

В параметре forwarders приводится список IP-адресов компьютеров, являющихся переадресующими серверами. Они опрашиваются в указанном порядке. При использовании переадресующего сервера традиционная процедура поиска домена (сначала опрашивается корневой сервер, затем осуществляется проход по цепочке отсылок) на-

[14] Некоторые брандмауэры умеют распознавать ответ DNS, поступивший на запрос, выданный секунду тому назад. Такому брандмауэру этот параметр не нужен.

рушается, поэтому нужно внимательно следить за тем, чтобы не возникало циклов переадресации.

Сервер, для которого установлен атрибут `forward only`, опрашивает переадресующие серверы и кэширует их ответы самостоятельно. Если переадресующий сервер не ответит, запрос потерпит неудачу. Сервер с атрибутом `forward first` связывается с переадресующими серверами, но при необходимости сможет обработать запрос самостоятельно.

Поскольку у параметра `forwarders` нет значения по умолчанию, переадресация происходит только в том случае, когда она задана явно. Включать переадресацию можно либо глобально, либо в пределах отдельных зон.

```
allow-query { список_соответствия_адресов };          [все хосты]
allow-transfer [ список_соответствия_адресов ];       [все хосты]
blackhole [ список_соответствия_адресов ];            [пусто]
```

Эти параметры позволяют указать, какие хосты (или сети) могут обращаться к серверу имен и запрашивать пересылку зонных баз данных. В списке `blackhole` перечислены серверы, которые никогда не "удостоятся внимания" демона `named`: он не будет принимать от них запросы и обращаться к ним за ответом.

```
sortlist { список_соответствия_адресов }; [не должен использоваться]
```

Мы упоминаем этот параметр лишь для того, чтобы читатели помнили: его не нужно использовать. Изначально он был предназначен для оказания помощи примитивным распознавателям, которые неправильно сортируют наборы записей. В списке `sortlist` можно задать точный порядок, в соответствии с которым будут возвращаться множественные ответы, что противоречит внутренней логике текущих версий пакета BIND.

Среди других параметров, которые нарушают привычный ход событий, следует упомянуть параметр `rrset-order`, определяющий, в какой последовательности должны возвращаться множественные ответы: циклической, фиксированной или случайной. Имеется также параметр `topology`, посредством которого можно попытаться предугадать, как будет осуществляться выбор удаленных серверов. В большинстве случаев ни один из этих параметров использовать не нужно.

```
lame-ttl число;          [600 (10 минут)]
max-ncache-ttl число;    [10800 (3 часа)]
max-cache-ttl число;     [604800 (1 неделя)]
```

Параметры времени жизни (TTL) устанавливают лимиты значений, контролирующих помещение в кэш положительных и отрицательных результатов запросов, а также безрезультатных запросов. Все входящие записи ресурсов поступают с уже заданными значениями времени жизни, однако эти параметры позволяют локальному серверу установить собственные ограничения, чтобы сократить объем памяти, расходуемой на кэшированные данные. Параметры `max-` задают значения времени жизни и никогда не увеличивают их. В спецификациях DNS сказано, что время жизни для отрицательных ответов (например, "no such domain") должно быть меньше времени жизни настоящих данных.

Параметр `lame-ttl` определяет, в течение какого времени (задается в секундах) в кэше будет находиться уведомление о неправильно сконфигурированном сервере. Максимальным является значение, равное 1800 секундам, или 30 минутам. Большинство неправильно сконфигурированных серверов (см. раздел 15.16) остаются таковыми в течение неопределенного периода времени, так как их владельцы не знают, что их серверы

не синхронизированы с родительской зоной. Если этому параметру присвоить максимальное значение, то все должно быть в порядке.

Параметр `max-ncache-ttl` задает лимит времени жизни для отрицательных ответов, находящихся в кэше. По умолчанию он равен трем часам, а максимальное значение соответствует одной неделе. Параметр `max-cache-ttl` задает максимальное время жизни для положительного ответа на запрос.

```
enable-dnssec yes | no;      [no]
```

Этот параметр конфигурирует поддержку DNSSEC. Сейчас он по умолчанию равен `no`, хотя есть надежда, что через год-другой он будет равен `yes`.

```
ends-udp-size число;         [4096]
```

Раньше для пакетов UDP по умолчанию устанавливался размер 512 байт; каждый компьютер, подключенный к сети Internet, должен иметь возможность заново собирать фрагментированный пакет из 512 или менее байтов. Несмотря на то что этот лимит имел смысл в 80-х гг. прошлого столетия, сейчас в нем уже нет необходимости. Сейчас этот лимит серьезно мешает программам, таким как серверы имен, работающим с UDP для запросов и ответов.

В системе BIND 9 размер пакета UDP по умолчанию равен 4096 байтам (максимально допустимый размер). Этого достаточно, чтобы уместить новые особенности, такие как DNSSEC, IPv6 и интернационализированные доменные имена. Тем не менее, некоторые брандмауэры не пропускают пакеты UDP, размер которых больше 512 байт; если вы работаете "за спиной" такого брандмауэра, вам нужно отключить этот параметр и откорректировать работу брандмауэра. Допустимыми являются значения, попадающие в диапазон 512–4096 байт.

```
ixfr-from-differences yes | no;         [no]
```

Инкрементные зонные пересылки (incremental zone transfer — IXFR) ранее поддерживались только для тех зон, в которых использовались динамические обновления (разговор о них пойдет в разделе 15.14). Благодаря этому параметру зоны, поддерживаемые вручную (например, с помощью текстового редактора), теперь могут использовать преимущества инкрементных зонных пересылок. Инкрементные зонные пересылки экономят полосу пропускания сети за счет того, что часть ресурсов ЦП и памяти отводится на сортировку зонных записей, вычисления и применения различий. Этот параметр использует такую же устойчивую схему журнализации, что и при динамических обновлениях.

```
root-delegation-only exclude {сисок_имен};[пусто]
zone "имя" {type delegation-only}
```

Инструмент Site Finder, разработанный сетевой компанией VeriSign, перенаправляет пользователей, допустивших опечатки при вводе адреса, на собственные серверы с помощью записи DNS метасимвола. В свое время были предприняты попытки избавиться от этого своеобразного пиратства средствами политики, хотя в то же время в BIND 9 были реализованы в качестве полумер тип зоны `delegation-only`, параметр зоны `delegation-only` и глобальный параметр `root-delegation-onl`. Эти параметры ограничивают корневые серверы и серверы TLD в возможности предоставления информации о делегировании (записи серверов имен и адреса серверов имен) и запрещают им предоставлять данные об хостах. Если это будет разрешено, ваши локальные кэширующие серверы будут возвращать сообщение "no such domain" в ответ на неверно введенный запрос.

📖 Сведения об инструменте Site Finder можно найти в разделе 15.6.

Предложение exclude было добавлено по той причине, что не все TLD являются зонами только для делегирования. Например, “museum”, “us” и “de” не являются таковыми, а “com”, “net” и “org” должны быть таковыми. В следующем фрагменте, взятом из зонных файлов домена cs.colorado.edu, показан пример использования этого нового типа зоны для нейтрализации работы инструмента Site Finder.

```
// предназначено для нейтрализации VeriSign Site Finder
zone "com" { type delegation-only; };
zone "net" { type delegation-only; };
```

Инструкция acl

Список управления доступом — это просто именованный список соответствия адресов:

```
acl имя_списка {
    список_соответствия_адресов
};
```

Заданное имя можно указывать везде, где требуется список соответствия адресов.

Инструкция acl должна быть самой первой в файле named.conf, так что не пытайтесь ставить ее среди других объявлений. Файл named.conf читается за один проход, поэтому списки управления доступом должны быть определены до того, как на них встретится ссылка. Существуют четыре стандартных списка:

- any — соответствует всем хостам;
- localnets — соответствует всем хостам локальной сети;
- localhost — соответствует самому компьютеру;
- none — не соответствует ни одному хосту.

Сети, входящие в группу localnets, определяются адресами интерфейсов компьютера с учетом сетевых масок.

Инструкция key

Инструкция key определяет именованный ключ шифрования, используемый для аутентификации на сервере. Подробнее о механизмах аутентификации, применяемых в пакете BIND, речь пойдет в разделе 15.15. Здесь же мы лишь вкратце опишем суть процесса.

Для создания ключа нужно указать как алгоритм шифрования, так и совместный секретный ключ, который представлен в виде строки, закодированной в формате Base64 (данный алгоритм кодирования описан в документе RFC1521):

```
key идентификатор_ключа {
    algorithm строка;
    secret строка;
};
```

Как и в случае списков управления доступом, идентификатор ключа должен быть определен с помощью инструкции key в файле named.conf до того, как встретится ссылка на ключ. Чтобы связать ключ с конкретным сервером, включите идентификатор ключа в список keys соответствующей инструкции server. Ключ используется как для проверки запросов, поступающих от сервера, так и для подписи ответов на эти запросы.

Инструкция trusted-keys

Инструкция trusted-keys является частью механизма аутентификации DNSSEC, описанного в документе RFC2535. Каждая запись состоит из пяти компонентов, определяющих имя домена, флаги, протокол, алгоритм шифрования и ключ. Все это необходимо для безопасного взаимодействия с сервером имен домена. Формат инструкции таков:

```
trusted-keys {
   домен флаги протокол алгоритм ключ;
   домен флаги протокол алгоритм ключ;
   ...
};
```

Каждая строка представляет ключ конкретного домена. Атрибуты *флаги*, *протокол* и *алгоритм* являются неотрицательными целыми числами. Атрибут *ключ* — это строка, закодированная в формате Base64.

Инструкция trusted-keys используется в тех случаях, когда зона имеет цифровую подпись, а ее родительская зона — нет, поэтому нельзя быть уверенным в том, что открытый ключ, получаемый от DNS-сервера, действительно надежен.

Подробнее о технологии DNSSEC рассказывается в разделе 15.15.

Инструкция server

Демон named способен общаться с серверами, которые не используют последнюю версию пакета BIND или просто неправильно сконфигурированы. Инструкция server сообщает демону характеристики удаленных серверов.

```
server IP-адрес {
   bogus yes | no;                                [no]
   provide-ixfr yes | no;                         [yes]
   request-ixfr yes | no;                         [yes]
   edns yes | no;                                 [yes]
   transfers число;                               [2]
   transfer-format one-answer | many-answers;     [many-answers]
   keys { ключ; ключ; ... };
   transfer-source IP-адрес [порт]
   transfer-source-v6 IP-адрес_v6 [порт]
};
```

С помощью инструкции server можно переопределять значения глобальных конфигурационных параметров, относящихся к серверам. Просто укажите нужные параметры и их новые значения.

Если для сервера задан атрибут bogus, демон named не посылает ему запросы. Этот атрибут следует устанавливать для серверов, которые работают неправильно. Атрибут bogus отличается от глобального параметра blackhole тем, что он запрещает только исходящие запросы. В отличие от него, параметр blackhole полностью исключает все формы связи с перечисленными серверами.

Сервер имен в BIND 9, являющийся главным сервером зоны, выполняет инкрементные зонные пересылки, если атрибут provide-ixfr равен yes. Точно так же подчиненный сервер запрашивает инкрементные зонные пересылки, если атрибут request-ixfr равен yes.

Параметр `edns` определяет, будет ли локальный сервер использовать расширенный протокол DNS при общении с удаленным сервером. Многие новые функциональные возможности пакета BIND (например, поддержка стандарта IPv6 и спецификация DNSSEC) вызывают генерирование пакетов, размер которых превышает 512 байт, что является пределом для UDP-пакетов. Следовательно, они могут применяться лишь в сочетании с протоколом EDNS0.

Подчиненные серверы синхронизируются с главным сервером путем получения зонной пересылки во время изменения данных (подробно о зонных пересылках можно прочитать в разделе 15.14). Параметр `transfers` ограничивает количество одновременных входящих зонных пересылок от удаленного сервера. Это серверный эквивалент глобального параметра `transfers-in`, но поскольку он применяется только к одному серверу, то, по сути, переопределяет параметр `transfers-per-ns`. Его имя изменено для сохранения совместимости с BIND 8.

Параметр `transfer-format` является эквивалентом одноименного глобального параметра. Его приходится использовать при взаимодействии серверов BIND 8/9 и BIND 4 или когда в сети есть серверы имен Microsoft. NT-системы обрабатывают только пересылки, состоящие из одной записи. В Windows 2000 подобное ограничение отсутствует, хотя в ранних версиях нельзя было работать с сообщениями DNS, размер которых превышал 16 Кбайт.

В списке `keys` задаются идентификаторы ключей, которые ранее были определены в инструкции `key` для использования в сигнатурах транзакций TSIG (о них рассказывается в разделе 15.15). К любому запросу, посылаемому удаленному серверу, будет добавлена сигнатура, зашифрованная с помощью этого ключа. Запросы, поступающие от удаленного сервера, могут не иметь цифровой подписи, но если она есть, то будет проверена.

Предложения `transfer-source` предоставляют адрес (IPv4 или IPv6) интерфейса (и, необязательно, порт), который нужно использовать в качестве исходного адреса (порта) запросов зонной пересылки. Это предложение требуется только тогда, когда в системе имеется множество интерфейсов, а в предложении `allow-transfer` удаленного сервера задан некоторый IP-адрес.

Инструкция `masters`

Инструкция `masters` позволяет указать один или несколько главных серверов с помощью IP-адресов и ключей шифрования. Указанное имя затем можно использовать в параметре `masters` инструкций `zone`, чтобы не повторять IP-адреса и ключи.

📖 Разве может быть несколько главных серверов? Об этом — далее в разделе.

Использовать средство `masters` удобно, когда несколько подчиненных или усеченных зон получают свои данные из одних и тех же удаленных серверов. Если адреса или ключи шифрования удаленных серверов будут изменены, вы сможете обновить инструкцию `masters`, вместо того чтобы изменять различные инструкции `zone`.

Синтаксис инструкции:

```
masters имя {IP-адрес [port IP-адрес_порта] [key ключ]; ...};
```

Инструкция logging

Демон named в настоящее время заслуживает награды "За наиболее конфигурируемую подсистему журнальной регистрации на Земле". Система Syslog предоставляет программистам контроль над приоритетами сообщений, а системным администраторам — контроль над местом хранения этих сообщений. Но в рамках заданного приоритета администратор не может указать: "Это сообщение меня интересует, а это — нет". В BIND 8 были добавлены категории, позволяющие классифицировать сообщения по типам, и каналы, расширяющие возможности в плане хранения сообщений. Категории назначаются программистом, а каналы — администратором.

Поскольку вопросы журнальной регистрации лежат несколько в стороне от нашего повествования (особенно если учесть объем материала), мы рассмотрим их позднее, в разделе 15.16.

Инструкция zone

Инструкции zone являются "сердцем" файла named.conf. Они сообщают демону named о зонах, для которых он авторитетен, и задают параметры управления каждой зоной. Инструкция zone также используется для предварительной загрузки "подсказок" с корневого сервера — имен и адресов корневых серверов, участвующих в инициализации DNS-поиска.

Точный формат инструкции zone зависит от роли, которую демон named должен играть в отношении этой зоны. Чтобы быть точнее, скажем, что возможными типами зон являются master, slave, hint, forward, stub и delegation-only.

Тип delegation-only рассматривался в разделе 15.11. Остальные будут рассмотрены в последующих кратких разделах.

Многие из рассмотренных ранее глобальных параметров могут быть частью инструкции zone, т.е. переопределять глобальные установки. Мы не будем повторно упоминать о них, за исключением тех параметров, которые наиболее часто используются.

Конфигурирование главного сервера зоны

Ниже показан формат инструкции zone для зоны, в которой демон named является главным сервером имен:

```
zone "имя_домена" {
   type master;
   file "путь";
};
```

Доменное имя в спецификации зоны всегда дается в двойных кавычках.

Зонная база данных хранится на диске в текстовом файле, доступном для редактирования. Поскольку нет соглашения об именовании этого файла, в объявлении зоны должна присутствовать директива file. Зонный файл представляет собой набор записей о DNS-ресурсах; их формат описан в разделе 15.7.

Часто задается также ряд других серверных атрибутов, например:

```
allow-query { список_соответствия_адресов };     [все хосты]
allow-transfer { список_соответствия_адресов }; [все хосты]
allow-update { список_соответствия_адресов };   [none]
zone-statistics yes | no;                        [no]
```

Параметры, связанные с управлением доступом, не являются обязательными. Если для зоны поддерживаются динамические обновления, в параметре `allow-update` должен содержаться список хостов, от которых разрешено принимать данные. Динамические обновления применимы только в отношении главных зон; параметр `allow-update` не может присутствовать в объявлении подчиненной зоны. Убедитесь, что в списке указаны лишь локальные DHCP-серверы[15].

Параметр `zone-statistics` заставляет демон `named` отслеживать статистику запросов/ответов, например точное число и общий процент отсылок, рекурсивных и ошибочных запросов. Конкретные примеры приведены в конце раздела 15.15.

При наличии столь большого числа зонных параметров (есть еще порядка тридцати, о которых мы не рассказали) конфигурация зоны кажется довольно сложной. На самом деле допускается объявление главной зоны, состоящее из имени зонного файла. В BIND 4 больше ничего нельзя задать. Ниже показан пример, взятый из документации и слегка модифицированный нами:

```
zone "example.com" {
   type master;
   file "forward/example.com";
   allow-query { any; };
   allow-transfer { my-slaves; };
};
```

Имя `my-slaves` относится к списку управления доступом, определенному ранее.

Конфигурирование подчиненного сервера зоны

Формат инструкции `zone` для подчиненного сервера будет почти таким же, как и в случае главного сервера:

```
zone "имя_домена" {
   type slave | stub;
   file "путь";
   masters { IP-адрес [port IP-адрес порта] [key имя_ключа]; ...};  [нет
значений по умолчанию]
   allow-query { список_соответствия_адресов };          [все хосты]
   allow-transfer { список_соответствия_адресов };       [все хосты]
};
```

Подчиненные серверы обычно хранят полную копию зонной базы данных. Но если тип сервера `stub`, а не `slave`, ему передаются только записи NS (описания серверов имен). Усеченные зоны позволяют демонам `named` родительской зоны автоматически узнавать, какие компьютеры предоставляют услуги DNS для дочерних зон. Это делается на тот случай, если администратор дочерней зоны забудет проинформировать родительскую зону об изменениях. Родительской зоне нужна эта информация, чтобы правильно делать отсылки либо рекурсивные запросы.

Директива `file` задает имя локального файла, в котором сохраняется реплицированная база данных. Получив новую копию данных зоны, сервер сохраняет ее в этом файле. В случае сбоя и перезагрузки сервера файл можно просто прочитать с диска, а не пересылать по сети.

[15] Необходима также входная фильтрация на брандмауэре, а еще лучше использовать технологию аутентификации TSIG.

Редактировать файл реплицированной базы данных не нужно, так как он управляется демоном named. Тем не менее, имеет смысл просмотреть его, если есть подозрение, что в базу данных главного сервера закралась ошибка. Файл подчиненной зоны создается уже после того, как демон named интерпретировал исходные данные зоны и раскрыл все относительные имена. Когда в файле присутствуют имена наподобие

```
128.138.243.151.cs.colorado.edu.
anchor.cs.colorado.edu.cs.colorado.edu.
```

можно быть уверенным в том, что где-то пропущена завершающая точка.

В списке masters перечислены IP-адреса компьютеров, с которых может быть загружена зонная база данных. В нем может быть также имя списка главных серверов, определенных в предыдущем списке masters.

Выше мы говорили о том, что лишь один компьютер может быть главным сервером зоны. Почему же можно указывать более одного адреса? По двум причинам. Во-первых, у главного компьютера может быть несколько сетевых интерфейсов и, соответственно, несколько IP-адресов. Когда один из интерфейсов становится недоступным (проблемы с сетью или маршрутизацией), остаются в резерве другие интерфейсы. Следовательно, рекомендуется указывать все топологически различные адреса главного сервера.

Во-вторых, демону named в действительности все равно, откуда поступают данные зоны. Он так же легко загружает базу данных с подчиненного сервера, как и с главного. Этой особенностью демона можно воспользоваться для того, чтобы сделать подчиненный сервер, с которым легко устанавливается связь, резервной копией главного сервера. В любом случае IP-адреса проверяются по очереди до тех пор, пока не будет найден работающий сервер. Теоретически можно даже создать иерархию серверов, в которой главный сервер обслуживает несколько серверов второго уровня, а те, в свою очередь, обслуживают множество серверов третьего уровня.

Задание корневых "подсказок"

Инструкция zone типа hint сообщает демону named местонахождение файла, из которого он может загрузить имена и адреса корневых серверов имен, чтобы предварительно заполнить свой кэш:

```
zone "." {
   type hint;
   file "путь";
};
```

"Подсказки" — это набор DNS-записей, в которых перечислены серверы корневого домена ("."). Они нужны для того, чтобы демон named знал, откуда начинать поиск доменов. Не имея "подсказок", демон знал бы только о тех доменах, которые сам обслуживает, а также об их поддоменах.

Когда named начинает свою работу, он перезагружает "подсказки" из одного из корневых серверов. Следовательно, у вас все будет в порядке, если в вашем файле "подсказок" будет хотя бы один действительный, досягаемый корневой сервер.

В BIND 9 "подсказки" корневого сервера встроены в скомпилированный код пакета, поэтому конфигурировать корневую зону не требуется. Но если все же задать файл "подсказок", то пакет примет его. Мы рекомендуем предоставлять явные "подсказки". Сегодня в сферу DNS начала вторгаться политика, поэтому корневые серверы имен и их IP-адреса стали больше подвержены изменениям.

Файл "подсказок" чаще всего называется `root.cache`. В нем содержатся результаты ответов, которые можно получить, запросив у корневого сервера список серверов имен в корневом домене. Фактически, вы можете сгенерировать файл "подсказок" с помощью команды `dig`:

```
% dig @f.root-servers.net . ns > root.cache
```

Обратите внимание на аргумент-точку. Если сервер f.root-servers.net не отвечает, можно вообще не указывать конкретный корневой сервер:

```
% dig . ns > root.cache
```

В результате список корневых серверов будет получен из кэша локального сервера имен, а не из авторитетного источника. Но ничего страшного в этом нет — даже если сервер имен не перезагружался или не перезапускался год или два, он периодически сам обновлял записи корневых серверов по мере их устаревания.

Вот пример файла "подсказок" (взято из `dig@f.root-servers.net . ns`):

```
; <<>> DiG 9.3.0rc1 <<>> @f.root-servers.net . ns
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 28797
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUESTION SECTION:
;.                      IN    NS
;; ANSWER SECTION:
.            518400    IN    NS    I.ROOT-SERVERS.NET.
.            518400    IN    NS    J.ROOT-SERVERS.NET.
.            518400    IN    NS    K.ROOT-SERVERS.NET.
.            518400    IN    NS    L.ROOT-SERVERS.NET.
...
;; ADDITIONAL SECTION:
I.ROOT-SERVERS.NET. 3600000    IN    A    192.36.148.17
J.ROOT-SERVERS.NET. 3600000    IN    A    192.58.128.30
K.ROOT-SERVERS.NET. 3600000    IN    A    193.0.14.129
L.ROOT-SERVERS.NET. 3600000    IN    A    198.32.64.12
...
;; Query time: 3 msec
;; SERVER: 192.5.5.241#53(f.root-servers.net)
;; WHEN: Wed Jul 7 13:54:26 2004
;; MSG SIZE rcvd: 436
```

Обратите внимание на точки, с которых начинаются записи первого набора. Это не пятна, оставленные типографским станком. На самом деле они определяют домен (корень), к которому относятся записи NS.

Задание зоны переадресации

Зона типа `forward` переопределяет глобальные параметры переадресации демона `named` (описаны в разделе 15.11) для конкретного домена:

```
zone "имя_домена" {
   type forward
   forward only | first;
   forwarders { IP-адрес; IP-адрес; ...}
};
```

Создавать такую зону имеет смысл, если у организации имеется деловой партнер и нужно направлять трафик непосредственно его серверам имен в обход стандартного пути распространения запросов.

Инструкция controls

Инструкция controls определяет, каким образом команда rndc будет управлять работой демона named. Эта команда может запускать и останавливать демон, выводить отчет о его состоянии, переводить демон в режим отладки и т.д. Будучи сетевой утилитой, она требует должной конфигурации, чтобы злоумышленники не могли через Internet влиять на работу сервера имен. Формат инструкции controls таков:

```
controls {
   inet IP-адрес port IP-адрес_порта allow { список_соответствия_адресов }
               keys { список_ключей };
}
```

Если параметр ports опущен, то по умолчанию rndc будет использовать порт 953 для обмена данными с named.

Удаленное управление сервером имен кажется одновременно удобной и опасной процедурой. В BIND 9 необходимо включить принудительную аутентификацию посредством аргумента *ключ* предложения allow. При этом ключи, указанные в списке соответствия адресов, игнорируются, а используются только те ключи, которые перечислены в списке keys.

В BIND 9 существует команда rndc-confgen, позволяющая сгенерировать ключ аутентификации, используемый в диалоге между командой rndc и демоном named. Это можно сделать двумя способами: заставить обе стороны загрузить ключ из общего конфигурационного файла (/etc/rndc.key) или поместить ключ в два разных конфигурационных файла (/etc/rndc.conf для rndc и /etc/named.conf для named). Второй способ сложнее, но он необходим, если демон named и команда rndc запускаются на разных компьютерах.

В BIND 9 при отсутствии инструкции controls в список соответствия адресов заносится адрес интерфейса обратной связи, а ключ ищется в файле /etc/rndc.conf. Поскольку аутентификация в этой версии пакета обязательна, команда rndc не сможет управлять работой демона named без ключа. Это кажется чрезмерным требованием, но подумайте вот о чем: даже если команда rndc работает только на хосте localhost (127.0.0.1) и его адрес заблокирован для внешнего мира на брандмауэре, вы все равно выдаете всем локальным пользователям определенный кредит доверия, подразумевая, что они не будут делать ничего противозаконного с сервером имен. В то же время любой из них может подключиться с помощью утилиты telnet к управляющему порту и ввести “stop”.

Ниже показаны результаты работы команды rndc-confgen, генерирующей 256-битный ключ. Размер ключа объясняется тем, что он позволяет уместить результаты на странице. Обычно вывод команды направляется не на экран, а в файл /etc/rndc.conf. Комментарии в нижней части включают строки, которые необходимо добавить в файл named.conf, чтобы демон named и команда rndc могли взаимодействовать.

```
% ./rndc-confgen -b 256
# Start of rndc.conf
key "rndc-key" {
  algorithm hmac-md5;
```

```
    secret "orZuz5amkUnEp52zlHxD6cd5hAC1dOGsG/elP/dv2IY=";
};

options {
    default-key "rndc-key";
    default-server 127.0.0.1;
    default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list
# as needed:
# key "rndc-key" {
#    algorithm hmac-md5;
#    secret "orZuz5amkUnEp52zlHxD6cd5hAC1dOGsG/elP/dv2IY=";
# };
#
# controls {
#    inet 127.0.0.1 port 953
#      allow { 127.0.0.1; } keys { "rndc-key"; };
# };
# End of named.conf
```

Автоматический режим, в котором генерируется совместный конфигурационный файл, проще. Выполняемая от имени пользователя `root`, команда `rndc-confgen -a -b 256` создает файл `/etc/rndc.key`, содержащий следующие данные:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "laGbZj2Cobyc0m/jFVNCu8OJzsLKNH+CCb2JCWY6yJw=";
};
```

Этот файл имеет режим доступа 600 и должен принадлежать пользователю `named` либо `root`.

Впоследствии его можно будет включить в конфигурационный файл `named.conf` с помощью директивы `include`.

Расщепление DNS и инструкция `view` в BIND 9

Во многих организациях требуется, чтобы внутреннее представление сети отличалось от представления этой сети с точки зрения пользователей Internet. Например, внутренние пользователи могут видеть все компьютеры зоны и лишь несколько разрешенных внешних серверов. Или другой вариант: внутреннее и внешнее представления одинаковы по охвату хостов, однако внутренним пользователям предоставляются дополнительные (либо отличающиеся) записи. В частности, записи MX, предназначенные для маршрутизации электронной почты, за пределами домена ссылаются на почтовый концентратор, а внутри домена — на отдельные рабочие станции.

📖 О частных адресных областях рассказывалось в разделе 12.4.

Расщепленная конфигурация особенно удобна организациям, которые во внутренних сетях применяют IP-адреса из частного диапазона (RFC1918). К примеру, запрос об имени хоста с IP-адресом 10.0.0.1 не может быть обработан глобальной системой доменных имен, но он вполне допустим в контексте локальной сети. Среди запросов, поступающих на корневые серверы имен, 4-5% либо имеют исходный IP-адрес из частного

диапазона, либо касаются такого адреса. Ни на один из этих запросов нельзя ответить. Запросы обоих типов являются следствием неправильной конфигурации либо пакета BIND, либо "доменов" Microsoft.

Инструкция `view` содержит список адресов, определяющий, кто из клиентов имеет доступ к данному представлению, а также ряд параметров, применимых ко всем зонам в представлении, и, наконец, определения самих зон. Синтаксис инструкции таков:

```
view имя_представления {
   match-clients { список_соответствия_адресов };
   параметр_представления; ...
   определение_зоны; ...
};
```

В списке `match-clients` перечислены клиенты представления. Представления просматриваются по порядку, поэтому инструкции `view` с самыми большими ограничениями доступа должны идти впереди. Зоны в разных представлениях могут иметь одинаковые имена, а принимать свои данные могут из разных файлов. Представления налагают ограничение на структуру файла `named.conf`: если они присутствуют, *все* инструкции `zone` должны находиться в контексте представлений.

Ниже приводится взятый из документации к BIND 9 пример. В обоих представлениях задается одна и та же зона, но с разной базой данных:

```
view "internal" {
   match-clients { наши сети; };  // только внутренние сети
   recursion yes;                 // только для внутренних клиентов
   zone "example.com" {           // полное представление зоны
      type master;
      file "example-internal.db";
   };
};

view "external" {
   match-clients { any; };        // допускаются любые запросы
   recursion no;                  // рекурсия запрещена
   zone "example.com" {           // только "общедоступные" хосты
      type master;
      file "example-external.db";
   };
};
```

Если поменять порядок представлений, никто не сможет получить доступ к внутреннему представлению. Адреса внутренних хостов пройдут проверку на соответствие условию `any` в предложении `match-clients` внешнего представления до того, как начнет проверяться внутреннее представление.

Во втором примере конфигурации DNS, показанном в следующем разделе, будут даны некоторые дополнительные примеры представлений.

15.12. Примеры конфигурации пакета BIND

После подробного знакомства с файлом `named.conf` настало время рассмотреть ряд готовых примеров. В следующих подразделах мы опишем три типичные конфигурации:

- зона localhost;
- сервер небольшой компании, занимающейся вопросами безопасности и применяющей расщепленную конфигурацию DNS;
- эксперты: `isc.org`, консорциум Internet Systems Consortium.

Зона localhost

Адрес 127.0.0.1 обозначает сам хост и должен преобразовываться в имя "localhost."[16]. Одни сайты преобразовывают адрес в "localhost.*локальный_домен*.", а другие делают и то, и другое. Соответствующим адресом (IPv6) является ::1.

Если вы забудете сконфигурировать зону localhost, ваш сайт будет запрашивать у корневого сервера информацию о localhost. Корневые серверы получают настолько много таких запросов, что операторы подумывают о добавлении обобщенного преобразования между localhost и 127.0.0.1 на корневом уровне. Измерения, производимые на корневом сервере F в Сан-Франциско в марте 2004 г., показали, что localhost был третьим по популярности доменом после "arpa" и "com". Понятно, что загруженному серверу имен приходится иметь дело с бесполезными запросами.

Прямое преобразование в имя localhost.*локальный_домен* может быть определено в файле зоны прямого преобразования самого домена или его собственного файла. Каждый сервер, даже кэширующий, обычно является главным сервером по отношению к его собственному обратному домену localhost.

Ниже представлены строки конфигурационного файла `named.conf`, в которых задается конфигурация localhost:

```
zone "localhost" {// зона прямого преобразования localhost
   type master;
   file "for/localhost";
   allow-update { none; };
};

zone "0.0.127.in-addr.arpa" {// зона обратного преобразования localhost
type master;
file "rev/127.0.0";
allow-update { none; };
};
```

Соответствующий файл зоны прямого преобразования, `for/localhost`, содержит следующее:

```
$TTL 30d
; localhost.
@          IN   SOA   localhost. postmaster.localhost. (
                         1998050801; Порядковый номер
                         3600      ; Период обновления
                         1800      ; Интервал между попытками
                         604800    ; Интервал устаревания
                         3600 )    ; Минимальное время жизни
                NS   localhost.
                A    127.0.0.1
```

[16] На самом деле, к localhost относится вся сеть 127/8 класса A, однако многие пользователи просто используют 127.0.0.1.

а файл зоны обратного преобразования, `rev/127.0.0`, содержит:

```
$TTL 30d
; 0.0.127.in-addr.arpa
@          IN  SOA   localhost. postmaster.localhost. (
                        1998050801 ; Порядковый номер
                        3600       ; Период обновления
                        1800       ; Интервал между попытками
                        604800     ; Интервал устаревания
                        3600 )     ; Минимальное время жизни
                NS   localhost.
1               PTR  localhost.
```

Преобразование в локальный адрес (127.0.0.1) никогда не изменяется, поэтому таймауты можно сделать более продолжительными. Обратите внимание на порядковый номер, в котором зашифрована дата; последнее изменение файла датируется 1998 г. Обратите внимание также на то, что для домена localhost указан только главный сервер имен. Знак `@` здесь следует понимать как “0.0.127.in-addr.arpa.”.

Небольшая компания, предоставляющая консалтинговые услуги в области безопасности

Наш первый реальный пример посвящен небольшой компании, специализирующейся на предоставлении консультаций по компьютерной безопасности. На ее сервере Red Hat Enterprise Linux (последняя версия сервера) установлен пакет BIND 9 и применяется расщепленная конфигурация DNS, в которой внутренние и внешние пользователи получают разную информацию. В компании используются адреса из частного диапазона. Запросы, касающиеся этих адресов, не должны выходить в Internet и засорять глобальную систему доменных имен. Ниже приведен соответствующий файл `named.conf`, слегка переформатированный и снабженный комментариями.

```
options {
    directory "/var/domain";
    version "root@atrust.com";
    allow-transfer {82.165.230.84; 71.33.249.193; 127.0.0.1; };
    listen-on { 192.168.2.10; 192.168.2.1; 127.0.0.1; 192.168.2.12; };
};

include "atrust.key";              // файл с режимом доступа 600,
                                   // содержащий определение ключа "atkey"

controls {
   inet 127.0.0.1 allow { 127.0.0.1; } keys { atkey; };
};

view "internal" {                  // внутреннее представление

    match-clients { 192.168.0.0/16; 206.168.198.192/28; 172.29.0.0/24; };
    recursion yes;

    zone "." IN {                  // зона корневых "подсказок"
       type hint;
       file "named.cache";
    };
```

```
    zone "localhost" IN {              // зона прямого преобразования
                                       // хоста localhost
       type master;
       file "localhost.forw";
       allow-update { none; };
    };
    zone "0.0.127.in-addr.arpa" IN {   // зона обратного преобразования
                                       // хоста localhost
       type master;
       file "localhost.rev";
       allow-update { none; };
    };
    zone "atrust.com" IN {             // внутренняя зона
                                       // прямого преобразования
       type master;
       file "internal/atrust.com";
    };
    zone "1.168.192.in-addr.arpa" IN { // внутренняя зона
                                       // обратного преобразования
       type master;
       file "internal/192.168.1.rev";
       allow-update { none; };
    };                                 // множество других зон обратного
                                       // преобразования здесь опущены
    zone "atrust.us" {                 // подчиненный сервер во
                                       // внутреннем представлении
       type slave;
       file "internal-slave/atrust.us";
       masters { 127.0.0.1; };
    };
    zone "atrust.org" {
       type slave;
       file "internal-slave/atrust.org";
       masters { 127.0.0.1; };
    };                                 // множество других зон обратного
                                       // преобразования здесь не приводится
};                                     // конец внутреннего представления

view "world" {                         // внешнее представление

    match-clients { any; };
    recursion no;

   // определения зон для доменов "." и localhost аналогичны
   // показанным выше, поэтому опущены для наглядности

    zone "atrust.com" {            // внешняя зона прямого преобразования
       type master;
       file "world/atrust.com";
       allow-update { none; };
    };
    zone "189.173.63.in-addr.arpa" { // внешняя зона обратного преобразования
       type master;
       file "world/63.173.189.rev";
       allow-update { none; };
    };
```

```
    zone "atrust.us" {            // главный сервер в представлении master
       type master;
       file "world/appliedtrust.com-aggregate";
       allow-update { none; };
    };

    zone "atrust.org" {           // заметьте, что файл один и тот же
       type master;
       file "world/appliedtrust.com-aggregate";
       allow-update { none; };
    };                            // множество зон atrust
                                  // здесь не приводится

    zone "admin.com" {            // главные зоны в представлении world
       type master;
       file "world/admin.com";
       allow-update { none; };
    };                            // множество главных зон
                                  // здесь не приводится
    zone "eldoraskipatrol.org" {  // подчиненные зоны только
                                  // в представлении world
   type slave;
   file "world-slave/eldoraskipatrol.org";
   masters { 192.231.139.1; };
};                                // множество подчиненных зон
                                  // здесь не приводится
};                                // конец внешнего представления
```

Файл `atrust.key` содержит определение ключа “atkey”:

```
key "atkey" {
   algorithm hmac-md5;
   secret "совместный секретный ключ";
};
```

Зоны организованы в соответствии с их представлением и типом (главный/подчиненный); эту же схему отражает и условие именования зонных файлов данных. Этот сервер является рекурсивным для внутреннего представления, которое включает все локальные хосты, многие из которых используют частную адресацию. Сервер не является рекурсивным для внешнего представления, которое содержит только выбранные хосты в домене artust.com и внешние зоны, для которых они предоставляют главную или подчиненную службу DNS.

Зонные файлы `internal/atrust.com` и `world/atrust.com` определяют два разных представления. Ниже показаны их фрагменты. Вот текст внутреннего представления:

```
; atrust.com — внутренний файл
$TTL 86400
$ORIGIN com.
atrust              3600  SOA  ns1.atrust.com. trent.atrust.com. (
                                   2004012900 10800 1200 3600000 3600 )
                    3600  NS   NS1.atrust.com.
                    3600  NS   NS2.atrust.com.
                    3600  MX   10 mailserver.atrust.com.
                    3600  A    66.77.122.161
```

```
$ORIGIN atrust.com.
ns1                         A   192.168.2.11
ns2                         A   66.77.122.161
www                         A   66.77.122.161
mailserver                  A   192.168.2.11
exchange                    A   192.168.2.100
secure                      A   66.77.122.161
bark                        A   192.168.2.10
superg                      A   192.168.1.249
at-dmz-gw                   A   192.168.1.254
at-external-gw              A   192.168.2.254
at-external-outside-gw      A   206.168.198.220
indras-gw                   A   206.168.198.222
; dhcp host range
dhcp-0-hardwire          IN   A   192.168.1.64
dhcp-1-hardwire          IN   A   192.168.1.65
; ...

; "поддомен" booklab; заметьте, что это поддомен, а не подзона
redhat.booklab            IN   A   192.168.10.1
redhat-ent.booklab        IN   A   192.168.10.2
debian.booklab            IN   A   192.168.10.3
fedora.booklab            IN   A   192.168.10.4
freebsd.booklab           IN   A   192.168.10.5
suse.booklab              IN   A   192.168.10.6
macos.booklab             IN   A   192.168.10.7
solaris.booklab           IN   A   192.168.10.8
```

А вот текст внешнего представления той же зоны `world/atrust.com`:

```
; atrust.com — внешний файл
$TTL 57600
$ORIGIN .
atrust.com                SOA   ns1.atrust.com. trent.atrust.com. (
                                  2004020400 10800 1200 3600000 3600 )
                          NS    NS1.atrust.com.
                          NS    NS2.atrust.com.
                          MX    10 mailserver.atrust.com.
                          A     66.77.122.161
ns1.atrust.com.           A     206.168.198.209
ns2.atrust.com.           A     66.77.122.161

$ORIGIN atrust.com.
www                       A     66.77.122.161
mailserver                A     206.168.198.209
bark                      A     206.168.198.209
secure                    A     66.77.122.161

; обратные преобразования
exterior1                 A     206.168.198.209
209.198.168.206           PTR   exterior1.atrust.com.
exterior2                 A     206.168.198.213
213.198.168.206           PTR   exterior2.atrust.com.
exterior3                 A     206.168.198.220
220.198.168.206           PTR   exterior3.atrust.com.
exterior4                 A     206.168.198.210
210.198.168.206           PTR   exterior4.atrust.com.
```

Обратите внимание: когда одному компьютеру требуется несколько имен, они представляются с помощью дополнительных записей A, а не CNAME. Во внешнем представлении world видимыми являются лишь некоторые хосты.

Параметр TTL в этих зонных файлах равен 16 часам (57 600 секунд). Для внутренних зон он равен одному дню (86 400 секунд). Для большинства записей зонных файлов значение TTL не назначается явно. Оно — необязательное и может быть добавлено в любую строку перед названием типа записи.

Загадочные записи PTR в конце файла внешнего представления позволяют провайдеру компании atrust.com делегировать ей полномочия по управлению обратными привязками в очень маленьких областях адресного пространства. Это делается с помощью записей CNAME на хосте провайдера. Подробнее о специальном применении записей CNAME рассказывается в разделе 15.7.

Internet System Consortium, isc.org

BIND — детище консорциума ISC. Этот консорциум занимается сопровождением системы BIND и является оператором корневого сервера имен F. У них есть также сервер TLD, обслуживающий около 60 доменов верхнего уровня. Вот почему мы называем членов этого консорциума экспертами!

Ниже представлены фрагменты конфигурационных файлов isc.org. Обратите внимание на то, что в них используются протоколы IPv4 и IPv6. Кроме того, в них применяется схема шифрования с использованием TSIG, позволяющая аутентифицировать главные и подчиненные серверы при зонных пересылках. Параметры `transfer-source` гарантируют, что исходные IP-адреса для исходящих запросов зонных преобразований согласованы с инструкциями `allow-transfer` главных серверов.

Конфигурационный файл `named.conf`:

```
// сервер имен isc.org TLD

options {
    directory "/var/named";
    datasize 1000M;
    listen-on { 204.152.184.64; };
    listen-on-v6 { 2001:4f8:0:2::13; };
    recursion no;
    transfer-source 204.152.184.64;
    transfer-source-v6 2001:4f8:0:2::13;
};

// ключ rndc
key rndc_key {
    algorithm hmac-md5;
    secret "<secret>";
};

// Ключ TSIG для сервера имен ns-ext
key ns-ext {
    algorithm hmac-md5;
    secret "<secret>";
};
```

```
server 204.152.188.234 { keys { ns-ext; }; };

controls {
   inet 204.152.184.64 allow { any; } keys { rndc_key; };
};

include "inf/named.zones";    // корневой, localhost, 127.0.0.1, ::1
include "master.zones";       // зоны, которыми мы управляем
include "slave.zones";        // подчиненные зоны
```

Использование инструкций `include` позволяет сделать файл `named.conf` кратким и аккуратным. При обслуживании большого числа зон конфигурационный файл можно разбить на несколько небольших частей, как это сделано в этом файле. Более того, нужно настроить иерархию файловой системы таким образом, чтобы у вас не оказалось каталога с тысячами зонных файлов. Современным файловым системам Linux приходится иметь дело с большими каталогами, и нужно сказать, что они эффективно справляются с этой задачей, хотя в плане управления этими системами возникают трудности.

Ниже представлен фрагмент конфигурационного файла `master.zones`:

```
zone "isc.org" { // ISC
   type master;
   file "master/isc.org";
   allow-update { none; };
   allow-transfer { none; };
};

zone "sfo2.isc.org" { // ISC
   type master;
   file "master/sfo2.isc.org";
   allow-update { none; };
   allow-transfer { none; };
};
// множество зон было опущено в целях наглядности
```

А это фрагмент файла `slaves.zones`:

```
zone "vix.com" {
   type slave;
   file "secondary/vix.com";
   masters { 204.152.188.234; };
};

zone "cix.net" {
   type slave;
   file "secondary/cix.net";
   masters { 204.152.188.234; };
};
```

Теперь, когда вы уже достаточно хорошо знакомы с DNS и конфигурацией BIND, мы можем перейти к рассмотрению более практических задач, связанных с запуском службы имен на сайте, включая поддержку зонных файлов, обеспечения безопасности, тестирования и отладки.

15.13. Запуск демона named

Наличие системы BIND в дистрибутивах Linux зависит как от самого дистрибутива, так и от параметров, выбранных во время инсталляции дистрибутива. Если BIND не был инсталлирован, вам придется получить и инсталлировать пакет BIND. В пакете содержится также сценарий запуска демона `named`, который запускается с помощью `init`: `/etc/init.d/named` для RHEL, Fedora и SUSE, и `/etc/init.d/bind9` для Debian и Ubuntu.

Демон `named` запускается во время загрузки системы и работает вместе с ней в параллельном режиме. Для управления работающей копией `named` используется интерфейс командной строки `rndc` (см. раздел 15.16).

Сценарии демона `init`, запускающие демон `named` в каждой из систем, имеют дополнительные точки входа (например, `reload`), которые предназначены для системных администраторов. Но их реализация в ряде дистрибутивов достаточно спорная, поэтому проще все же пользоваться командой `ndc`. Мы рекомендуем не вмешиваться в работу сценариев запуска, операционной системы и демона `init`; применяйте `rndc` после того, как демон `named` начнет свою работу.

Демон `named` регистрирует журнальные сообщения в системе Syslog, поэтому демон `syslogd` должен запускаться раньше. Не используйте для управления сервером имен демон `inetd` или `xinet`; при малейшей необходимости он будет перезапускать демон `named`, замедляя время его реакции и препятствуя накоплению полезной информации в кэше. В некоторых сайтах используется постоянно работающий сценарий, который помещает команду запуска `named` в бесконечный цикл, чтобы защитить `named` от "смерти" (пример этого можно найти в разделе 15.9). Дистрибутив BIND включает сценарий `nanny`, играющий похожую роль.

15.14. Обновление файлов зон

Когда в домен вносится изменение (например, добавляется или удаляется компьютер), следует обновить файлы данных на главном сервере. Кроме того, необходимо увеличить порядковый номер в записи SOA для этой зоны и выполнить команду `rndc reload`, которая просигнализирует демону `named` о необходимости принять изменения. Можно просто уничтожить и перезапустить демон, но это приведет к удалению кэшированных данных, полученных от других доменов.

В первых версиях пакета BIND для управления демоном `named` применялись сигналы и стандартная системная команда `kill`, но когда у разработчиков перестало хватать сигналов, появилась команда `rndc`, которая исправила ситуацию. В BIND 9 сигналы уже не поддерживаются (за исключением сигнала HUP, вызывающего повторное чтение конфигурационного файла, перезагрузки зон и перезапуска `named`, и сигналов TERM и INT, приводящих к завершению работы демона), поэтому рекомендуем пользоваться командой `rndc`.

Обновленные данные немедленно передаются подчиненным серверам, поскольку параметр `notify` включен по умолчанию. Если же по какой-то причине он отключен, подчиненным серверам придется дождаться, пока истечет период обновления, установленный в записи SOA зоны (обычно от одного до шести часов). Если необходима более срочная корректировка, можно выполнить на подчиненном сервере команду `rndc reload`, которая заставит демон связаться с главным сервером, убедиться в том, что зонная база данных изменена, и запросить ее пересылку.

При изменении имени или IP-адреса компьютера не забывайте модифицировать зоны как прямого, так и обратного преобразований. Игнорирование последних может привести к появлению скрытых ошибок: часть команд будет работать, а часть — нет.

Если изменить файлы данных, но не обновить порядковый номер в записи SOA, то изменения вступят в силу только на главном сервере (после перезагрузки), но не на подчиненных.

Не следует редактировать файлы данных, относящиеся к подчиненным серверам. Эти файлы ведет сам демон `named`, и системные администраторы не должны вмешиваться в его работу. Можно просматривать файлы данных, не внося в них никаких изменений.

Документом RFC2136 разрешается менять зонные базы данных программным путем. Эта возможность, называемая динамическим обновлением, необходима для поддержки протоколов автоматического конфигурирования, например DHCP. О том, как работает этот механизм, речь пойдет чуть ниже.

Зонные пересылки

Серверы DNS синхронизируются посредством механизма зонных пересылок. В исходной спецификации DNS (и в BIND 4) требовалось, чтобы все данные, относящиеся к той или иной зоне, пересылались одновременно. Инкрементные обновления были определены в документе RFC1995 и реализованы в BIND 8.2.

Когда требуется обновить информацию, подчиненный сервер запрашивает пересылку с главного сервера и создает резервную копию данных зоны на диске. Если данные на главном сервере не изменились, что можно определить путем сравнения порядковых номеров (а не самих данных), то обновления не происходит, и резервные копии файлов остаются практически теми же (лишь время их модификации устанавливается равным текущему времени).

Зонные пересылки осуществляются по протоколу TCP через порт 53. Соответствующее событие регистрируется системой Syslog с пометкой “transfer-*”. Организация IETF определила, что инкрементные обновления могут осуществляться по протоколу TCP либо UDP, но в BIND реализован лишь первый вариант.

В процессе зонной пересылки и передающий, и принимающий серверы доступны для запросов. Подчиненный сервер начинает использовать новые данные только по завершении операции. В BIND 8 пересылка осуществляется путем вызова специальной утилиты `named-xfer`, а в BIND 9 демон `named` управляет операцией напрямую.

Если зона очень велика (например, “com”) или обновляется динамически (о чем пойдет речь в следующем подразделе), то объем изменений обычно мал в сравнении с размером зоны. В этом случае используются инкрементные обновления, т.е. пересылаются только изменения (когда размер изменений начинает превышать размер зоны, включается режим обычной пересылки). Этот механизм напоминает программу `patch`: старая база данных сравнивается и синхронизируется с новой.

В BIND 9 инкрементные пересылки поддерживаются по умолчанию для любой зоны, сконфигурированной на прием динамических обновлений, а демон `named` самостоятельно ведет журнал транзакций, называющийся *`имя_зоны`*`.jnl`. В инструкции `server` теперь поддерживаются два отдельных параметра: `provide-ixfr` и `request-ixfr`. Первый из них включает или отключает механизм инкрементных пересылок для тех зон, где сервер является главным. Второй — делает то же самое для тех зон, где сервер является подчиненным.

```
provide-ixfr yes ;          # в инструкции server (BIND 9)
request-ixfr yes ;          # в инструкции server (BIND 9)
```

Начиная с версии BIND 9.3, демон `named` может предоставлять инкрементные зонные пересылки подчиненным серверам, чтобы те могли организовывать пересылку изменений (правок), произведенных вручную. Для этой цели нужно воспользоваться параметром `ixfr-from-differences` (см. раздел 15.11).

Для инкрементной зонной пересылки нужно, чтобы зонный файл был отсортирован в каноническом порядке. Об этом заботится демон `named`, однако в силу требований нужно использовать инкрементные пересылки. Чтобы снизить сетевую нагрузку, в процессе обмена на серверах расходуется больше ресурсов памяти и ЦП.

Немало усилий было потрачено на то, чтобы в случае сбоя сервера во время инкрементной пересылки зонные базы данных не превращались в "мусор". Запрос на инкрементную пересылку к серверу, который не поддерживает этот механизм, будет автоматически преобразован в обычный запрос на обновление.

Динамическое обновление

Система DNS основана на предпосылке, что соответствия между именами и адресами относительно стабильны и меняются нечасто. Но это правило постоянно нарушается, если в организации используется протокол DHCP и при подключении к сети компьютеру динамически назначается IP-адрес. Существуют два классических решения этой проблемы: добавить обобщенные записи в базу данных DNS или непрерывно редактировать DNS-файлы. В большинстве случаев ни одно из решений не является удовлетворительным.

Первое решение должно быть знакомо каждому, кто имеет коммутируемый выход в Internet. Конфигурация DNS в этом случае выглядит примерно следующим образом:

```
dhcp-host1.domain.    IN  A   192.168.0.1
dhcp-host2.domain.    IN  A   192.168.0.2
```

Это простое решение, но оно означает, что указанные имена постоянно связаны с IP-адресами, и, следовательно, компьютер, получающий новый адрес, меняет имя. В такой среде трудно соблюдать требования безопасности и вести журнальную регистрацию.

Механизм динамических обновлений, появившийся в последних версиях BIND, предлагает альтернативное решение. Он позволяет демону DHCP уведомлять сервер BIND о сделанных адресных назначениях, обновляя, таким образом, содержимое базы данных DNS "на лету". С помощью этого механизма можно добавлять, удалять и модифицировать записи о ресурсах. Во время динамических обновлений ведется журнал транзакций (*имя_зоны*`.jnl`), необходимый на случай сбоя сервера.

Динамически обновленную зону нельзя редактировать вручную, не остановив прежде поток динамического обновления. Это можно сделать с помощью команд `rndc freeze` *зона* или `rndc freeze` *представление_класса_зоны*. Эти команды синхронизируют журнальный файл с файлом главной зоны на диске, после чего удаляют журнальный файл. После этого файл зоны можно будет редактировать вручную. Естественно, первоначальный формат файла зоны будет утерян — файл будет выглядеть подобно файлам, поддерживаемым демоном `named` для подчиненных серверов.

При "замораживании" зоны динамическое обновление прекращается; с помощью команды `rndc thaw` и тех же аргументов, которые вы использовали для "замораживания", можно перезагрузить файл зоны с диска и возобновить динамические обновления.

Утилита nsupdate, входящая в дистрибутив BIND 9, позволяет осуществлять динамические обновления из командной строки. Утилита работает в пакетном режиме, принимая команды, вводимые с клавиатуры, или читая их из файла. Пустая строка или строка send обозначает конец обновления и посылает изменения на сервер. Две пустые строки обозначают конец входного потока. В командном языке предусмотрена примитивная инструкция if, позволяющая формулировать условия вида "если имя неизвестно в DNS, добавить его". Можно потребовать, чтобы искомое имя (или набор записей о ресурсах) существовало либо отсутствовало.

Ниже показан простейший сценарий, который заносит в базу данных информацию о новом хосте, а также псевдоним существующего хоста при условии, что этот псевдоним нигде не используется. Приглашение, обозначаемое угловыми скобками, генерируется утилитой nsupdate и не является частью командного сценария.

```
% nsupdate
> update add newhost.cs.colorado.edu 86400 A 128.138.243.16
>
> prereq nxdomain gypsy.cs.colorado.edu
> update add gypsy.cs.colorado.edu CNAME evi-laptop.cs.colorado.edu
```

Динамические обновления — очень опасная возможность. Они потенциально способны предоставить право неконтролируемой записи важных системных данных. Не пытайтесь контролировать доступ на основании IP-адресов: их легко подделать. Лучше воспользоваться системой аутентификации TSIG с совместным секретным ключом. BIND 9 поддерживает две команды:

```
% nsupdate -k каталог_ключа:файл_ключа
```

а в BIND 9 имеется еще и такая команда:

```
% nsupdate -y имя_ключа:секретный_ключ
```

Так как пароль вводится в командной строке, любой пользователь, выполнивший в подходящий момент команду w или ps, сможет выведать его. Поэтому предпочтительным вариантом является использование ключа -k. Подробнее о технологии TSIG рассказывается в разделе 15.15.

Динамические обновления зоны разрешаются в файле named.conf посредством параметра allow-update или update-policy. Параметр allow-update предоставляет право обновления любых записей клиентам, чьи IP-адреса и ключи шифрования указаны в списке. Параметр update-policy появился в BIND 9 и позволяет точнее управлять обновлениями на основании имен хостов и типов записей. Он требует использовать механизмы аутентификации. Оба параметра являются параметрами зоны.

С помощью параметра update-policy можно разрешить клиентам обновлять свои записи A и PTR, но не SOA, NS или KEY. С его помощью можно также позволить хосту обновлять только свои записи. Имена допускается задавать явно, в виде поддоменов, с метасимволами или с помощью ключевого слова self, которое задает правила доступа компьютера к собственным записям. Записи о ресурсах идентифицируются по классу или типу. Синтаксис правила update-policy выглядит следующим образом:

```
update-policy ( grant | deny ) объект имя_типа имя [типы];
```

объект — это имя криптографического ключа, необходимого для авторизации обновления. *имя_типа* может принимать одно из четырех значений: name, subdomain, wildcard или self. *имя* — это зона, требующая обновления, а *типы* — типы записей о

ресурсах, которые могут быть обновлены. Если тип не будет указан, то будут обновлены все типы, кроме SOA, NS, RRSIG и NSEC. Приведем пример:

```
update-policy { grant dhcp-key subdomain dhcp.cs.colorado.edu A };
```

В такой конфигурации любому, у кого есть ключ `dhcp-key`, разрешается обновлять адресные записи в поддомене dhcp.cs.colorado.edu. Эта директива должна появиться в файле `named.conf` главного сервера в инструкции `zone` домена dhcp.cs.colorado.edu. Потребуется также инструкция `key`, содержащая определение ключа `dhcp-key`.

Ниже представлен фрагмент файла `named.conf`, позаимствованный у факультета вычислительной техники Университета Колорадо. В этом файле используется инструкция `update-policy`, которая разрешает студентам, занимающимся в классе системного администрирования, обновлять свои собственные поддомены, не мешая остальной среде DNS.

```
// saclass.net
zone "saclass.net" in {
    type master;
    file "saclass/saclass.net";
    update-policy {
        grant feanor_mroe. subdomain saclass.net.;
        grant mojo_mroe. subdomain saclass.net.;
        grant dawdle_mroe. subdomain saclass.net.;
        grant pirate_mroe. subdomain saclass.net.;
        // и так далее
};
...
```

15.15. Вопросы безопасности

В старые добрые времена сеть Internet была очень небольшой, дружественной и была полезна в основном только компьютерным фанатам. Сегодня это враждебная среда, объединяющая миллионы компьютеров во всем мире. Описываемые в этом разделе механизмы безопасности кому-то могут показаться чрезмерными и даже слегка параноидальными, но подозрительность и предусмотрительность — печальная необходимость в современном мире.

DNS появилась как совершенно открытая система, но в процессе развития она становилась все более защищенной или, по крайней мере, приобретала необходимые средства защиты. По умолчанию каждый, у кого есть доступ в Internet, может исследовать чужой домен отдельными запросами с помощью таких команд, как `dig`, `host` или `nslookup`. В некоторых случаях можно получить образ всей базы данных DNS.

Для устранения этих недостатков в последние версии пакета BIND были введены различные средства управления доступом, основанные на проверке адресов или криптографической аутентификации. В табл. 15.11 перечислены элементы подсистемы безопасности, которые настраиваются в файле `named.conf`.

Демон `named` способен функционировать в среде, где корневой каталог изменен с помощью команды `chroot`, и при этом иметь непривилегированный идентификатор пользователя, что позволяет минимизировать риск, связанный с неправильным применением демона. Благодаря сигнатурам транзакций демон может контролировать динамические обновления или зонные пересылки. И, конечно же, он полностью поддерживает спецификацию DNSSEC. Обо всем этом пойдет речь ниже.

Таблица 15.11. Средства защиты в файле named.conf

Параметр	Инструкции	Что определяет
acl	Разные	Списки управления доступом
allow-query	options, zone	Кто может посылать запросы зоне или серверу
allow-recursion	options	Кто может посылать рекурсивные запросы
allow-transfer	options, zone	Кто может запрашивать зонные пересылки
allow-update	zone	Кто может делать динамические обновления
blackhole	options	Какие серверы нужно полностью игнорировать
bogus	server	Какие серверы никогда нельзя опрашивать
update-policy	zone	Какие обновления разрешены

Еще раз о списках управления доступом

Список управления доступом — это именованный список соответствия адресов, который может служить аргументом различных директив, в частности `allow-query`, `allow-transfer` и `blackhole`. Синтаксис списков был рассмотрен при описании инструкции `acl` в разделе 15.11. Подобные списки позволяют усилить защиту DNS несколькими различными способами.

В каждой организации должен существовать хотя бы один список для недоступных адресов и один — для локальных. Например:

```
acl bogusnets {          // список недоступных и фиктивных сетей
   0.0.0.0/8;            // адреса по умолчанию
   1.0.0.0/8;            // зарезервированные адреса
   2.0.0.0/8;            // зарезервированные адреса
   169.254.0.0/16;       // канально-локальные делегируемые адреса
   192.0.2.0/24;         // тестовые адреса, наподобие example.com
   224.0.0.0/3;          // пространство групповых адресов
   10.0.0.0/8;           // частное адресное пространство (RFC1918)[17]
   172.16.0.0/8;         // частное адресное пространство (RFC1918)
   192.168.0.0/16;       // частное адресное пространство (RFC1918)
};

acl cunets {             // список сетей университета штата Колорадо
   128.138.0.0/16;       // основная кампусная сеть
   198.11.16.0/24;
   204.228.69.0/24;
};
```

Далее нужно в глобальной секции `options` конфигурационного файла разместить следующие директивы:

```
allow-recursion { cunets; };
blackhole { bogusnets; };
```

Желательно также ограничить зонные пересылки только легитимными подчиненными серверами. Это достигается с помощью следующих списков:

[17] Не делайте частные адреса недоступными, если они используются для конфигурирования внутренних DNS-серверов!

```
acl ourslaves {
   128.138.242.1;         // сервер anchor
   ...
};

acl measurements {
   198.32.4.0/24;         // проект Билла Маннинга, адрес v4
   2001:478:6:0::/48;     // проект Билла Маннинга, адрес v6
};
```

Собственно ограничение реализуется такой строкой:

```
allow-transfer { ourslaves; measurements; };
```

Пересылки разрешены только нашим подчиненным серверам, а также компьютерам двух глобальных исследовательских проектов, которые посвящены определению размеров сети Internet и процента неправильно сконфигурированных серверов. Подобное ограничение лишает остальных пользователей возможности получать дамп всей базы данных с помощью команды `dig` (см. раздел 15.16).

Например, на сервере caida.org используется список управления доступом для ограничения пересылок на вторичные серверы; команда `dig @`*сервер домен* `axfr` будет пропущена, если была сгенерирована на компьютере jungle.caida.org, и не будет пропущена, если была сгенерирована на компьютере gypsy.cs.colorado.edu:

```
jungle$ dig @rommie.caida.org caida.org axfr
; <<>> DiG 8.3 <<>> @rommie.caida.org caida.org axfr
; (1 server found)
$ORIGIN caida.org.
@               4H   IN   SOA   @ postmaster (
                                 200406300     ; Порядковый номер
                                 1H            ; Период обновления
                                 30M           ; Интервал между попытками
                                 1W            ; Интервал устаревания
)
...
```

```
gypsy$ dig @rommie.caida.org caida.org axfr

; <<>> DiG 9.2.4 <<>> @rommie.caida.org caida.org axfr
;; global options: printcmd
; Transfer failed.
```

По-прежнему нужно защищать сеть на более низком уровне с помощью списков управления доступом на маршрутизаторе и стандартных средств защиты на каждом хосте. Если такой возможности нет, ограничьте трафик DNS-пакетов шлюзовым компьютером, который находится под постоянным административным контролем.

Ограничение возможностей демона named

Чтобы предотвратить ущерб, который может быть нанесен в случае атаки на сервер, можно запустить демон `named` в среде с измененным корневым каталогом, запустить от имени непривилегированного пользователя, или запустить в каждом из этих режимов. Флаг `-t` задает корневой каталог демона, а флаг `-u` определяет значение UID, присваиваемое процессу `named`. К примеру, команда

```
# named -u 53 -t /var/named
```

запускает демон с идентификатором пользователя 53 и корневым каталогом /var/named.

Хакер, взломавший демон named, получает доступ к системе от имени пользователя, запустившего демон. Если это пользователь root и корневой каталог не был изменен, последствия могут оказаться разрушительными. Многие администраторы не заботятся об использовании флагов -u и -t, но тогда в случае обнаружения очередной "дыры" они должны быстрее ставить "заплаты", чем хакеры будут атаковать систему.

Новый корневой каталог не может быть пустым, так как должен содержать файлы, необходимые для нормальной работы демона named: /dev/null, зонные файлы, named.conf, целевые файлы Syslog, именованный сокет ndc и т.д. Демон named в системе BIND 9 выполняет системный вызов chroot после загрузки всех библиотек, поэтому теперь больше не нужно копировать библиотечные файлы в каталог chroot.

По умолчанию Fedora инсталлирует демон named в измененный корневой каталог.

Безопасные межсерверные взаимодействия посредством технологий TSIG и TKEY

Пока спецификация DNSSEC (описана в следующем подразделе) находилась в стадии принятия, IETF разработала более простой механизм, названный TSIG (RFC2845). Он позволял организовывать безопасное взаимодействие серверов благодаря использованию "сигнатур транзакций". Контроль доступа, основанный на таких сигнатурах, надежнее, чем контроль на основе только исходных IP-адресов. Механизм TSIG способен защитить зонные пересылки между главным сервером и подчиненными серверами, и может реализовать защищенные динамические обновления.

Сигнатурами TSIG подписываются сообщения и ответы между серверами, но не между сервером и распознавателем. Сигнатуры транзакций проверяют подлинность хоста и проверяют, не были ли сфальсифицированы данные. Сигнатура проверяется в момент поступления пакета и тут же отбрасывается; она не сохраняется в кэше и не заносится в базу данных DNS.

В сигнатурах транзакций применяется симметричная схема шифрования, т.е. ключ шифрования совпадает с ключом дешифрования. Такой ключ называется *совместным секретным ключом*. Несмотря на то что спецификация TSIG допускает применение различных методов шифрования, в BIND реализован лишь один из них: HMAC-MD5.

Для каждой пары серверов, между которыми хотят организовать защищенный канал связи, должен создаваться свой ключ. Спецификация TSIG гораздо менее затратна в вычислительном плане, чем шифрование с открытым ключом, но она подходит только для локальной сети, где число пар взаимодействующих серверов невелико. На глобальную сеть эта спецификация не распространяется.

Утилита dnssec-keygen[18], являющаяся частью пакета BIND, генерирует ключ для пары серверов. Например, чтобы сгенерировать совместный ключ для двух серверов, master и slave1, понадобится следующая команда:

```
# dnssec-keygen -a HMAC-MD5 -b 128 -n HOST master-slave1
```

Флаг -b 128 создаст 128-битный ключ. Будут получены два файла: Kmaster-slave1.+157+09068.private и Kmaster-slave1.+157+09068.key, где 157 — код

[18] В BIND 8 она называется dnskeygen.

алгоритма HMAC-MD5, а 09068 — случайное[19] число, используемое в качестве идентификатора ключа на случай, если у одной пары серверов есть несколько ключей. Файл `.private` выглядит примерно так:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: jxopbeb+aPc71Mm2vc9R9g==
```

Содержимое файла `.key` будет таким:

```
master-slave1. IN KEY 512 3 157 jxopbeb+aPc71Mm2vc9R9g==
```

Оба файла должны иметь режим доступа 600 и принадлежать пользователю `named`.

Обратите внимание на то, что после строки аргумента `master-slave1` была добавлена точка (“`.`”) как в обоих именах файлов, так и в содержимом файла `.key`. Это объясняется тем, что в остальном содержимом имена ключей должны быть полностью определенными доменными именами и, поэтому, должны заканчиваться точкой.

Вообще-то, файл `.key` не нужен: он создается потому, что утилита `dnssec-keygen` применяется также для получения пар открытых ключей, когда открытый ключ (файл `.key`) вставляется в зонный файл в виде записи KEY. 512 в записи KEY означает не размер ключа, а флаговый бит, который показывает, что эта запись является записью ключа DNS.

Не исключено, что все эти сложности могут вас сбить с толку, и вы не заметите самого главного: сгенерированный ключ представляет собой всего лишь длинное случайное число. Создать ключ можно вручную, записав ASCII-строку нужной длины (она должна делиться на 4) и сделав вид, что она закодирована в формате Base64. Способ создания ключа не важен. Главное, чтобы он существовал на обоих компьютерах.

Скопируйте ключ на оба сервера посредством команды `scp` или другим надежным способом. *Не используйте* утилиты `telnet` и `ftp` для копирования ключа: даже внутренние сети могут быть небезопасными. Ключ должен присутствовать в файле `named.conf` на обоих компьютерах. В связи с тем, что этот файл — общедоступен, а ключ — нет, поместите ключ в отдельный файл, который будет подключаться к файлу `named.conf` посредством инструкции `include`.

📖 Команда `scp` является частью пакета SSH, описанного в разделе 20.11.

Например, можно создать файл `master-slave1.tsig` и включить в него следующий фрагмент:

```
key master-slave1. {
  algorithm hmac-md5;
  secret "сгенерированный_ключ";
};
```

Режим доступ к файлу должен быть 600, а идентификатор его владельца — таким же, как у демона `named`. В файл `named.conf`, ближе к началу, нужно добавить такую строку:

```
include "master-slave1.tsig";
```

На данном этапе лишь определены ключи. Чтобы их можно было использовать для подписи и проверки обновлений, главному серверу понадобится ключ для пересылок, а подчиненный сервер должен будет уведомлять главный сервер с помощью инструкции `server` и директивы `keys`. Например, можно добавить строку

```
allow-transfer {key master-slave1. ;};
```

[19] Вообще говоря, оно не является ни случайным, ни даже псевдослучайным.

в инструкцию `zone` на главном сервере и строку

```
server IP-адрес_главного_сервера { keys { master-slave1. ;} ;};
```

в конфигурационный файл `named.conf` подчиненного сервера. Если главный сервер разрешает динамические обновления, он может также использовать ключ в своей директиве `allow-update` в инструкции `zone`.

Мы использовали обобщенные имена `master` и `slave1` для обозначения серверов и ключа. Если вы будете применять ключи TSIG во множестве зон, вы должны будете также включить зону в вашу схему именования.

Чтобы проверить конфигурацию TSIG, выполните команду `named-checkconf`, чтобы убедиться в правильности синтаксиса. Затем с помощью команды `dig` проверьте зонную пересылку (`dig@главный_сервер axfr`) из сервера `slave1` и какого-нибудь другого компьютера. В первом случае команда должна выполниться, а во втором — нет, причем вы должны будете увидеть сообщение “Transfer failed” (Сбой при пересылке). Чтобы полностью убедиться, что все в порядке, удалите директиву `allow-transfer` и выполните команду `dig` снова. На этот раз команда должна будет выполниться в обоих случаях. (Только не забудьте после этого снова включить директиву `allow-transfer`!) А напоследок увеличьте порядковый номер зоны на главном сервере и проверьте, были ли отражены изменения в журнальном файле подчиненного сервера.

Если вы впервые имеете дело с сигнатурами транзакций, запустите демон `named` на какое-то время на уровне отладки 1 (о режиме отладки рассказывается в разделе 15.16) и посмотрите, не будут ли выданы сообщения об ошибках. Старые версии пакета BIND не понимают подписанных сообщений и генерируют ошибки, иногда даже отказываясь загружать зонные базы данных.

TKEY — это механизм в BIND 9, позволяющий двум хостам автоматически генерировать совместный секретный ключ без телефонных звонков и удаленного копирования ключа. При этом используется алгоритм обмена ключами Диффи-Хеллмана, в котором каждая сторона создает случайное число, выполняет над ним определенные математические операции и посылает результат противоположной стороне. Полученное число заданным образом объединяется с имеющимся, и получается один и тот же ключ. Злоумышленник, прослушивающий линию, будет знать о содержании сеанса, но не сможет выполнить обратные математические преобразования[20]. К сожалению, еще не написан код, который бы помог извлечь настоящую пользу из TKEY и избавил от необходимости хранить ключ TSIG в конфигурационном файле; однако есть надежда, что его скоро напишут.

При использовании ключей TSIG и сигнатур транзакций между главным и подчиненными серверами нужно следить за тем, чтобы часы сервера были синхронизированы с NTP (Network Time Protocol — протокол сетевого времени). Если часы слишком отстают или спешат (более 5 секунд в BIND 9), проверка подписей не будет работать. Устранить эту проблему очень сложно.

Пока что невозможно использовать ключи TSIG для защиты соединений между серверами BIND и Microsoft. Microsoft использует TSIG нестандартным способом (GSS-TSIG) и не будет открывать подробности этой схемы, иначе она нарушит условия соглашения о неразглашении сведений. По этому вопросу сейчас ведутся переговоры.

Еще одним механизмом подписи транзакций между серверами или между динамическими обновителями и главным сервером является механизм SIG(0). Он использует

[20] В основе методики лежит тот факт, что в модульной арифметике возведение в степень — легкая операция, а найти логарифм, чтобы снять степень, почти невозможно.

открытый ключ шифрования; более подробно о нем можно прочитать в документах RFC2535 и RFC2931.

Технология DNSSEC

DNSSEC — это набор расширений DNS, позволяющих аутентифицировать источник данных зоны и проверить их целостность, используя шифрование с открытым ключом. Таким образом, DNSSEC позволяет DNS-клиентам задавать вопросы вида "Действительно ли данные зоны поступили от владельца зоны?" и "Те ли это данные, которые послал владелец?".

Технология DNSSEC основана на цепочке доверия: корневые серверы предоставляют подтверждающую информацию для доменов верхнего уровня, те — для доменов второго уровня и т.д. Параметр конфигурации BIND `trusted-keys` позволяет усовершенствовать этот процесс и защитить части дерева DNS до того, как будут защищены корневой домен и домены верхнего уровня.

В системах шифрования с открытым ключом используются два ключа: один шифрует (подписывает) сообщение, а другой дешифрует (проверяет) его. Публикуемые данные подписываются секретным "личным" ключом. Любой может проверить правильность цифровой подписи с помощью соответствующего "открытого" ключа, который свободно распространяется. Если открытый ключ позволил правильно расшифровать зонный файл, значит, зона была зашифрована искомым личным ключом. Суть в том, чтобы убедиться в подлинности открытого ключа, используемого для проверки. Такие системы шифрования позволяют одной из сторон поставить свою "подпись" на открытом ключе, передаваемом другой стороне, гарантируя легитимность ключа. Отсюда термин "цепочка доверия".

Данные, составляющие зону, слишком велики для шифрования с открытым ключом: процесс шифрования получится слишком медленным. Вместе с тем данные сами по себе не секретны, поэтому они просто пропускаются через функцию хеширования (к примеру, вычисляется контрольная сумма MD5), а результат подписывается (шифруется) секретным ключом зоны. Полученный зашифрованный хеш-код называется *цифровой подписью*, или *сигнатурой* зоны. Цифровые подписи добавляются к данным, подлинность которых они устанавливают, в виде записей RRSIG в подписанных зонных файлах.

Для проверки сигнатуры ее нужно дешифровать открытым ключом подписавшей стороны, "прогнать" данные через тот же алгоритм хеширования и сравнить вычисленный хеш-код с расшифрованным. В случае совпадения подписавшее лицо считается аутентифицированным, а данные — целостными.

В DNSSEC каждая зона имеет открытые и секретные ключи. Секретным ключом подписывается каждый набор ресурсов (т.е. каждый набор записей одного типа, относящихся к одному хосту). Открытый ключ используется для проверки сигнатур и включается в зонную базу данных в виде записи DNSKEY.

Администраторы родительских зон подписывают открытые ключи своих дочерних зон. Демон `named` проверяет подлинность записи DNSKEY дочерней зоны, сравнивая ее с сигнатурой родительской зоны. Для проверки подлинности последней демон обращается к родительской зоне более высокого уровня и т.д. вплоть до корневой зоны. Ее открытый ключ имеется в файле корневых "подсказок".

Прежде чем переходить к детальному описанию ключей и сигнатур, следует сказать несколько слов о текущем статусе технологии DNSSEC и ее влиянии на системных администраторов. Системным администраторам придется иметь дело с DNSSEC через

год-другой, а пока они заняты другими проблемами. Многим приложениям требуется база для использования открытых ключей, и DNS — один из кандидатов на роль такой базы. Однако возникает любопытный парадокс. Необходимо убедиться в безопасности системы доменных имен, прежде чем доверять ей свои ключи. Но для того чтобы защитить саму DNS, требуется готовая криптосистема с открытым ключом.

Недавние изменения в спецификациях DNSSEC сделали эту технологию еще более способной к развертыванию, чем раньше. В исходном варианте DNSSEC хранила копии подписанных ключей для дочерних зон в родительском и дочернем доменах, поэтому если в дочернем или родительском домене нужно было изменить свой ключ, то для этого требовалась постоянная связь. Так как каждый ключ имел множество копий, было неясно, какой копии можно доверять.

В документе RFC4034 был введен новый тип записи о ресурсах (DS, или delegation signer), ключи были разделены на ключ подписания ключа (key-signing key — KSK) и ключ подписания зоны (zone-signing key — ZSK), а также были изменены имена некоторых записей о ресурсах:

- KEY теперь называется DNSKEY — для ключей шифрования;
- SIG теперь называется RRSIG — подпись для набора записей о ресурсах;
- NXT теперь называется NSEC — она необходима для идентификации следующего элемента защиты в делегировании.

Записи KEY и SIG по-прежнему используются в схемах защиты SIG(0) и TSIG. DNSKEY, RRSIG и NSEC применяются вместе с DNSSEC, а NXT уже не используется. Изменения из SIG в сторону RRSIG и из NXT в NSEC являются незначительными; имена были изменены с той целью, чтобы не путать старое ПО с новой схемой.

Ключи, находящиеся в записи о ресурсах DNSKEY, могут быть или ключами KSK, или ключами ZSK. Различить их можно с помощью нового флага SEP (security entry point — точка входа защиты). Если 15-й бит поля флага имеет значение 1, значит, это ключ KSK, если 0 — это ключ ZSK. То есть, ключи KSK всегда являются нечетными, а ключи ZSK — четными, что будет удобным при их обработке в качестве десятичных чисел.

Запись DS появляется только в родительской зоне и показывает, что подзона является защищенной (подписанной). Еще она идентифицирует ключ, используемый дочерней зоной для самостоятельного подписания ее личного ключа RRset. Запись DS включает идентификатор ключа (5-значное число), алгоритм шифрования, тип дайджеста и дайджест записи открытого ключа, разрешенного (или используемого) для подписи записи о ресурсах ключа в дочерней зоне.

Вопрос изменения имеющихся ключей в родительской и дочерней зоне оставался сложным. Казалось, что для этой операции требуется взаимодействие и связь между родительской и дочерней зоной. Создание записи DS, использование отдельных ключей (KSK и ZSK), а также применение множества пар ключей помогли справиться с этой задачей.

Множество ключей можно сгенерировать и подписать таким образом, чтобы стал возможным плавный переход от одного ключа к другому. Дочерняя зона может изменить свои ключи ZSK, не уведомляя об этом родительскую зону; ей нужно координировать свои действия с родительской зоной лишь при изменении ключа KSK. Пока производится обмен ключами, старый ключ и новый ключ являются действительными; как только истечет срок использования кэшированных значений в сети Internet, старый ключ можно удалить.

В нынешних версиях BIND нет технологии OpenSSL, поэтому если вы захотите использовать DNSSEC, вам нужно будет найти пакет, включающий поддержку DNSSEC, или загрузить библиотеки SSL с сайта `www.openssl.org`, скомпилировав затем BIND с учетом поддержки шифрования (нужно указать параметр `--with-openssl` для `./configure`). Если не сделать этого, вы не сможете выполнить команду `dnssec-keygen`. Хотя она по-прежнему будет генерировать ключи TSIG, поэтому OpenSSL вам не понадобится.

Для создания и использования подписанных зон требуется пройти несколько этапов. Сначала нужно сгенерировать одну или несколько пар ключей для зоны. Например, с помощью команды

```
# dnssec-keygen -a DSA -b 768 -n ZONE mydomain.com.
```

будет сгенерирована 768-битная пара ключей с помощью алгоритма DSA, которая будет использоваться для подписания зоны `mydomain.com`. Можно воспользоваться следующими алгоритмами шифрования (в скобках указан диапазон размеров ключа): RSAMD5 (512–4096), RSASHA1 (512–4096), DH (128–4096) и DSA (512–1024, кратно 64). Если добавить флаг `-f KSK` (здесь он не используется), можно будет идентифицировать сгенерированный ключ в виде ключа KSK, установив его бит SEP.

Команда `dnssec-keygen` выведет следующую стандартную информацию:

```
Kmydomain.com.+003+50302
```

Здесь `Kmydomain.com` — это имя ключа, `003` — идентификатор алгоритма DSA, а `50302` — идентификатор ключа. Команда `dnssec-keygen` создает файл, содержащий открытый и секретный ключи:

```
Kmydomain.com.+003+50302.key      # Открытый ключ
Kmydomain.com.+003+50302.private  # Секретный ключ
```

Секретный ключ используется для подписания записей зоны, а открытый — для проверки цифровых подписей. Открытый ключ обычно указывается сразу же после записи SOA.

В идеале, секретный ключ из любой пары ключей должен храниться на каком-нибудь внешнем источнике, или хотя бы на компьютере, не подключенном к сети Internet. Эта мера предосторожности невыполнима для динамически обновляемых зон и не подходит для ключей ZSK, но отлично подходит для ключей KSK, которые, по всей видимости, “живут” довольно долго.

Понять смысл этих файлов, сгенерированных командой `dnssec-keygen`, сложно, не изучив их содержимое, поэтому этим мы сейчас и займемся.

Kmydomain.com.+003+50302.key

```
mydomain.com. IN DNSKEY 256 3 3
       BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1WU6D7O7/GiBXwxAsvpgH6
       sNXE3uwZVaQFxvDHfa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6zRBoO
       3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4yL8N6B32akBTiIMLp01FOJe
       xqLe6QrJVE21eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOsGRfOf4hy
       lOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTMVzTk
       qA4KJsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
       WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcK
       g2A3LHpmg1S59utmpxJa
```

Kmydomain.com.+003+50302.private

```
Private-key-format: v1.2
Algorithm: 3 (DSA)
Prime(p): tLZTYI1WU6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDHfa6amy3JSSilc
        RNfiiOs3LfoyZzUWOceVo6zRBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4y
        L8N6B32akB
Subprime(q): w5HLHyxHPoQnoRN9GPQ+Hi6toHU=
Base(g): TiIMLp01FOJexqLe6QrJVE21eXzRqC58TC25R6TPMoOH6cuue5w8eNphcs
        OsGRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTM
        VzTkqA4K
Private_value(x): GqcQz8K56CmUxgo6ERuyEWMLVME=
Public_value(y): JsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
        WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3L
        Hpmg1S59utmpxJa
```

Чтобы содействовать работе ключей, вы должны создать несколько пар ключей KSK и подписать их в вашей родительской зоне. После того как между вами и вашей родительской зоной будет установлена доверительная связь (ключ KSK, подписанный родительской зоной и включенный в запись DS, делегирующую вашу защищенную зону), вы сможете использовать эту связь, чтобы запустить процесс обмена ключами. Просто отправьте родительской зоне этот ключ, чтобы вас можно было распознать, вместе с дополнительными ключами KSK, которые вы хотите подписать и на которые указывают записи DS. После подписания новых ключей вы сможете удалить исходный ключ.

После того как родительская зона подпишет ваш ключ KSK и вставит его в вашу запись DNS, можно подписывать реальные зонные данные. Этот процесс происходит следующим образом: берется зонный файл и после каждого набора записей о ресурсах ставятся записи RRSIG (содержат реальные сигнатуры) и NSEC (используются для обработки отрицательных ответов).

Чтобы подписать зону, нужно использовать команду `dnssec-signzone`. Например, команда

```
# dnssec-signzone -o mydomain.com db.mydomain Kmydomain.com+003+50302
```

читает зонный файл `db.mydomain` и создает его подписанную версию `db.mydomain.signed`. Кроме того, эта команда создает два файла с наборами ключей:

```
keyset-mydomain.com. # Набор ключей
dsset-mydomain.com. # Защищенный набор ключей делегирования
```

Эти файлы используются для подписания зон и ключей, соответственно. Они содержат следующее:

keyset-mydomain.com.

```
$ORIGIN .
mydomain.com   172800   IN   DNSKEY   256 3 3 (
                        BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1W
                        U6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDH
                        fa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6z
                        RBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx
                        4yL8N6B32akBTiIMLp01FOJexqLe6QrJVE21
                        eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOs
                        GRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9
                        hjzskX64e0QeeENXRV8sFvTMVzTkqA4KJsBC
                        clVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn
```

```
                        0RO0mpH6ZUIlWaSIE1pAxaZ7ynD4hT1RB5br
                        2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3
                        LHpmg1S59utmpxJa
                        ) ; key id = 50302
```

dsset-mydomain.com.

```
mydomain.com. IN DS 50302 3 1 1B44471AFD5B4F4463BB3A0D7B66B6ABC018DA96
```

Любые файлы наборов ключей для дочерних зон в текущем каталоге также подписываются и внедряются в записи DS, которые делегируют защищенные подзоны серверам дочерней зоны и аутентифицируют их с помощью подписанных ключей. На подписание зоны может уйти много времени.

Если подписанная зона передается команде `dnssec-signzone` в качестве аргумента, то подписи любых записей, срок действия которых скоро истечет, будет обновлен. Близость завершения срока действия определяется следующим образом: три четверти всего времени периода действия. По умолчанию подписи действительны в течение 30 дней, хотя в командной строке `dnssec-signzone` можно указать другой период. Повторная подпись обычно приводит к изменениям, однако порядковый номер автоматически не изменяется (пока что), поэтому системным администраторам нужно обновлять порядковый номер вручную. Подчиненные серверы зоны должны будут выполнить операцию по зонной пересылке для синхронизации друг с другом.

В следующем примере показаны зонные данные до и после подписания[21]. Так выглядят зонные данные до подписания:

```
$TTL 172800 ; 2 days
@            IN  SOA    ns.mydomain.com. hostmaster.mydomain.com. (
                          2006081200  ; Порядковый номер
                          7200        ; Период обновления (2 часа)
                          3600        ; Интервал между попытками (1 час)
                          1728000     ; Интервал устаревания (2 недели 6 дней)
                          172800      ; Минимальное время жизни (2 дня)
                          )
             IN  NS     ns.cs.colorado.edu.

mydomain.com. IN  DNSKEY 256 3 3
  BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1WU6D7O7/GiBXwxAsvpgH6
  sNXE3uwZVaQFxvDHfa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6zRBoO
  3GTYpZ6efrFUackXKr9WsadC+4W+2fGx4yL8N6B32akBTiIMLp01FOJe
  lOwkb6T7etH//EQgfkLWqcwolVF9hjzskX64e0QeeENXRV8sFvTMVzTk
  qA4KJsBCclVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn0RO0mpH6ZUIl
  WaSIE1pAxaZ7ynD4hT1RB5br2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcK
  g2A3LHpmg1S59utmpxJa

anchor       IN  A      128.138.242.1
             IN  A      128.138.243.140
             IN  MX     10 anchor
             IN  MX     99 @
awesome      IN  A      128.138.236.20
...
```

[21] Мы старались упростить пример и использовали всего два хоста и их записи A и MX, однако для `dnssec-signzone` нужна настоящая зона, поэтому нам пришлось довавить записи SOA, NS и DNSKEY. Просим извинить нас за это.

А так они выглядят после подписания:

```
; File written on Thu Nov 11 17:41:25 2006
; dnssec_signzone version 9.3.0
mydomain.com. 172800 IN SOA ns.mydomain.com. hostmaster.mydomain.com. (
                          2006081200   ; Порядковый номер
                          7200     ; Период обновления (2 часа)
                          3600     ; Интервал между попытками (1 час)
                          1728000  ; Интервал устаревания (2 недели 6 дней)
                          172800   ; Минимальное время жизни (2 дня)
                          )
                          172800 RRSIG  SOA 3 2 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    ElneYxZ3g9JnKbXdnmPhKVWfd13JTU8ajOO
                                    5dQta2WeBAatNuWt8dQ= )
                          172800  NS    ns.cs.colorado.edu.
                          172800  RRSIG  NS 3 2 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    BLG6LRrXtRHRdRFtTOmlQsadOIefqHAq5Rid
                                    PHZ74vOl/UkEW6wY6VA= )
                          172800 NSEC anchor.mydomain.com. NS SOA RRSIG NSEC DNSKEY
                          172800  RRSIG  NSEC 3 2 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    BCz31GPChdQrmNrZypv4xxmXDCThZ0IlkEGL
                                    TSkf7Q+TmCDmAADxmBE= )
                          172800  DNSKEY 256 3 3 (
                                    BMORyx8sRz6EJ6ETfRj0Ph4uraB1tLZTYI1W
                                    U6D7O7/GiBXwxAsvpgH6sNXE3uwZVaQFxvDH
                                    fa6amy3JSSilcRNfiiOs3LfoyZzUWOceVo6z
                                    RBoO3GTYpZ6efrFUackXKr9WsadC+4W+2fGx
                                    4yL8N6B32akBTiIMLp01FOJexqLe6QrJVE21
                                    eXzRqC58TC25R6TPMoOH6cuue5w8eNphcsOs
                                    GRfOf4hylOwkb6T7etH//EQgfkLWqcwolVF9
                                    hjzskX64e0QeeENXRV8sFvTMVzTkqA4KJsBC
                                    clVzrDSLAsLZtYH4g6VvrMZHuQ5C/ArCIsdn
                                    0RO0mpH6ZUIlWaSIE1pAxaZ7ynD4hT1RB5br
                                    2KiyGTr27dHi7QS4vOW7oDDPI9+lwAcKg2A3
                                    LHpmg1S59utmpxJa
                                    ) ; key id = 50302
                          172800  RRSIG DNSKEY 3 2 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    BAgZDfk/YCOhVfuoyG5pgfyFCmsGqg4W7uuM
                                    Rm5eNP9Bn0EbBnuT6X0= )
                          anchor.mydomain.com. 172800 IN A 128.138.242.1
                          172800 IN A   128.138.243.140
                          172800 RRSIG  A 3 3 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    BIRtKW0Um7ItfbPqRew+jKo152WJh+4nHkmK
                                    1ePNxjsQWcgaKm5jiMU= )
                          172800 MX    10 anchor.mydomain.com.
                          172800 MX    99 mydomain.com.
                          172800 RRSIG  MX 3 3 172800 20061211164125 (
                                    20061111164125 50302 mydomain.com.
                                    BGtmN2u30y1pMDzstWGgWZfXB3lDlmy5W6DP
                                      t/8D31QpyYNBjJPb8J4= )
```

```
172800 NSEC  awesome.mydomain.com. A MX RRSIG NSEC
172800 RRSIG  NSEC 3 3 172800 20061211164125 (
         20061111164125 50302 mydomain.com.
         BArN6oES72gzFgQmBHL3NzlquMbDbLfpvj7J
         3CSb/c8U/bciWGXsV3Q= )
         awesome.mydomain.com.
172800 IN    A 128.138.236.20
172800  RRSIG  A 3 3 172800 20061211164125 (
         20061111164125 50302 mydomain.com.
         BJ/qWBgLgS/2N5CoXGnI4vs91SsyIBKKfoq9
         R+VsMpRmnVrSi1DU1n8= )
172800  NSEC   mydomain.com. A RRSIG NSEC
172800  RRSIG  NSEC 3 3 172800 20061211164125 (
         20061111164125 50302 mydomain.com.
         BKoByqF5wUceb2vc8H2uealgKrejH4VZ0S5m
         Q4KukWCUo2IAFX+msQ4= )
```

Что и говорить, подписанная зона выглядит уродливо. Обычно она в четыре раза больше исходной зоны, да и логическая последовательность записей нарушается. С практической точки зрения файл подписанной зоны не может быть нормально воспринят человеком. Более того: из-за наличия записей SIG и NXT ее уже нельзя редактировать вручную.

Запись RRSIG содержит много информации, а именно:

- тип подписываемого набора записей;
- используемый алгоритм (в данном случае DSA — код 3);
- значение TTL для подписанного набора записей;
- срок действия сигнатуры (в формате *ггггммддччссс*);
- время подписания (также в формате *ггггммддччссс*);
- идентификатор ключа (в данном случае 50302);
- имя подписывающей стороны (“mydomain.com.”);
- собственно цифровая подпись.

Чтобы можно было работать с подписанной зоной, найдите в файле `named.conf` зону `mydomain.com` и поменяйте в инструкции `zone` параметр `file` так, чтобы он указывал на файл `db.mydomain.signed`, а не `db.mydomain`. Вот и все!

Отрицательные ответы

Цифровые подписи прекрасно подходят для положительных ответов следующего вида: “Вот IP-адрес хоста `anchor.cs.colorado.edu`, а вот сигнатура, подтверждающая, что данные действительно поступили из домена `cs.colorado.edu` и эти данные корректны”. Но что делать с отрицательными ответами вроде “Нет такого хоста”? При таких ответах обычно не возвращаются подписанные записи.

В DNSSEC проблема решается за счет записей NSEC, в которых указывается следующая запись зоны, отсортированной в каноническом порядке[22]. Например, если после за-

[22] Это разновидность алфавитного порядка, при котором имена, стоящие выше в иерархии DNS, указываются первыми. Например, в зоне `cs.colorado.edu` имя `cs.colorado.edu` следует перед любым именем *хост*`.cs.colorado.edu`. В рамках одного уровня иерархии порядок сортировки — алфавитный.

писи хоста anchor.cs.colorado.edu идет запись хоста awesome.cs.colorado.edu и поступает запрос к хосту anthill.cs.colorado.edu, то в ответ будет получена запись NXT такого вида:

```
anchor.cs.colorado.edu.   IN  NSEC  awesome.cs.colorado.edu A MX NSEC
```

Она говорит о том, что сразу после имени "anchor" в зоне cs.colorado.edu следует имя "awesome.cs.colorado.edu", а имени "anchor" соответствует как минимум одна запись A, MX и NSEC. Запись NSEC возвращается также в том случае, когда хост существует, но запись запрашиваемого типа отсутствует. Если, скажем, запрашивается запись LOC хоста anchor, будет выдана запись NSEC, показанная выше. Из нее клиент узнает о том, что поддерживаются только записи A, MX и NSEC.

Последняя запись NSEC в зоне ссылается на первый хост. К примеру, запись NSEC для хоста zamboni.cs.colorado.edu ссылается на первую запись, т.е. на домен cs.colorado.edu:

```
zamboni.cs.colorado.edu.   IN  NSEC  cs.colorado.edu A MX NSEC
```

Благодаря записям NSEC можно осуществить полную зонную пересылку, запрашивая по одной записи за раз. Это может быть нежелательно с точки зрения безопасности.

Microsoft и DNS

В Windows записи SRV используются для поиска практически всего: серверов имен, принтеров, файловых систем и т.д. Реализуя записи SRV, компания Microsoft следовала спецификациям организации IETF, но способ вставки этих записей в базу данных DNS посредством защищенных динамических обновлений совершенно нестандартен. Microsoft использует разновидность сигнатур транзакций (спецификация GSS-TSIG), также основанную на совместном секретном ключе. Этот ключ можно получить через систему Kerberos из центра распределения ключей (Key Distribution Center — KDC). Вся прелесть в том, что в настоящий момент реализация Kerberos, используемая компанией Microsoft, несовместима с открытой версией Kerberos 5.

Если нужно работать с Windows и использовать записи SRV, придется отказаться от существующей системы Kerberos и запустить сервер Windows Kerberos. Для тех организаций, где сетевая инфраструктура давно налажена, это становится неразрешимой проблемой. Компания Microsoft делает вид, будто использует открытые протоколы, а на самом деле просто старается не потерять клиентов. Речь не идет о том, чтобы взаимодействовать с другими системами и тем более пустить конкурентов в свою вотчину. Надеемся, компания хотя бы документирует свои расширения, чтобы организация ISC смогла согласовать спецификации TSIG и GSS-TSIG.

Через неделю после того, как была выпущена операционная система Windows 2000, число запросов, адресуемых корневым серверам DNS, резко возросло. Проведенные исследования показали, что неправильно сконфигурированные системы Windows 2000 пытались динамически обновить корневые зоны или зоны верхнего уровня. В результате число UDP-запросов к корневому серверу A увеличилось более чем вдвое. Хуже того, когда запросы на обновление заканчивались неудачей, системы Win2K запрашивали запись KEY, чтобы выполнить аутентифицированное обновление. Получая отказ, они делали последнюю попытку: открывали TCP-соединение в надежде на аутентифицированное динамическое обновление. У корневого сервера нет возможности отвечать на миллионы запросов об установлении TCP-соединения.

Ошибки, влияющие на инфраструктуру Internet, являются серьезными, и могут сбить с толку, поэтому мы надеемся, что Microsoft как можно скорее устранит эту проблему. Сначала мы думали, что этот раздел будет содержать лишь историческую справку, однако позже мы пересмотрели сетевой трафик только что инсталлированной системы Windows XP с пакетом обновлений Service Pack 2. К счастью, обнаружилось, что он оказался таким же бескомпромиссным, как и всегда.

Проблема возникала чаще всего в зонах обратного преобразования в пространствах приватных IP-адресов, определение которых дано в документе RFC1918. Информация об этих адресах никогда не должна покидать локальную среду, однако Windows-компьютеры пытаются обновить их домен, а в случае неудачи они продолжают двигаться по дереву имен DNS, пока не достигнут корневых серверов.

Например, если сервер DHCP назначил адрес 192.168.1.10, Windows-компьютер попытается динамически обновить домен 1.168.192.in-addr.arpa, а затем in-addr.arpa и arpa.

Windows-системы, в которых не менялись настройки DNS, принятые по умолчанию, являются соучастниками непрекращающейся, массированной, медленной, непреднамеренной и распределенной атаки типа "отказ в обслуживании" на корневые серверы имен. Чтобы утихомирить системы Windows 2000 или XP и заставить их корректно взаимодействовать с серверами имен UNIX или Linux, выполните описанную ниже последовательность действий.

- Щелкните правой кнопкой мыши на значке My Network Places (Сетевое окружение) и выберите в появившемся меню пункт Properties (Свойства), чтобы отобразить окно Network and Dial-up Connections (Сетевые подключения).
- Щелкните правой кнопкой мыши на названии соединения и выберите пункт Properties; описываемую процедуру следует повторить для каждого имеющегося соединения.
- Выберите элемент Internet Protocol (TCP/IP) (Протокол Интернета (TCP/IP)) и щелкните на кнопке Properties.
- Щелкните на кнопке Advanced... (Дополнительно...) в нижней части страницы свойств.
- Перейдите на вкладку DNS.
- Снимите отметку с флажка Register this connection's address in DNS (Зарегистрировать адреса этого подключения в DNS) в нижней части страницы.
- Закройте все окна, последовательно щелкая на кнопках OK.

Чтобы сэкономить время, операторы корневых зон делегировали зоны обратного преобразования пространства адресов RFC1918 серверам `prisoner.iana.org`, `blackhole1.iana.org` и `blackhole2.iana.org`. Эти серверы являются авторитетными ("prisoner" — главный сервер, серверы "blackhole" — подчиненные) в этих зонах и перехватывают динамические обновления, генерируемые Windows-системами, которые хотят рассказать всему миру о том, что имя "mypc" должно быть связано с адресом 10.0.0.1. Серверы `iana.org` посылают данные на ближайший хост (метод рассылки "anycast") и имеют экземпляры у большинства крупных провайдеров услуг Internet, поэтому фиктивные сообщения об обновлении DNS, проникающие из локального сайта, не смогут внести сумятицу в работу корневых серверов в сети Internet.

Но и это еще не все. Если предпочтительные серверы DNS не отвечают на запрос в течение одной секунды, Windows-системы начинают быстро посылать множество од-

новременных запросов каждому известному им серверу DNS[23]. Другими словами, при первых признаках перегрузки сервера или сети Windows пытается сгенерировать максимально возможный объем трафика на сервере и в сети. Эту особенность можно отключить, отключив службу DNS Client (DNS-клиент), однако следует иметь в виду, что при этом вы заблокируете кэш локальной DNS каждого компьютера. Чтобы отключить службу DNS Client, выполните следующие действия:

- Откройте панель управления.
- Выполните двойной щелчок на значке Administrative Tools (Администрирование).
- В открывшемся окне выполните двойной щелчок на значке Computer Management (Управление компьютером).
- В левой части появившегося окна щелкните на значке "+" рядом с названием Services and Applications (Службы и приложения), чтобы раскрыть список.
- Выберите в левой части окна элемент Services (Службы), а в правой части выполните двойной щелчок на элементе DNS Client (DNS-клиент).
- Выберите в списке Startup type (Тип запуска) элемент Manual (Вручную).
- Щелкните на кнопке Stop (Остановить), чтобы остановить выполнение DNS-клиента.
- Закройте все окна, последовательно щелкая на кнопках OK.

Если вы не используете службу Microsoft Active Directory, то вам, пожалуй, лучше всего будет использовать серверы UNIX/Linux для DNS и не пользоваться средствами Windows. К сожалению, Active Directory сопровождается всевозможными сложностями, хотя нужно сказать, что вы можете поддерживать ее с помощью BIND.

И последнее. Серверы Microsoft поднимают много шума, когда запрос направляется в зону, все серверы которой сконфигурированы неправильно (имеет место напрасное делегирование). Если ваша организация в свое время передала полномочия серверам, связь с которыми теперь утеряна, инсталлируйте пустой зонный файл. Это "успокоит" серверы Microsoft, и они перестанут терроризировать своими запросами корневые серверы и серверы верхнего уровня.

15.16. Тестирование и отладка

Демон `named` располагает встроенными средствами отладки, основное из которых — великолепная подсистема журнальной регистрации. Можно также задавать уровни отладки в строке вызова демона или устанавливать их посредством команды `rndc`. Имеются команды, заставляющие демон вывести статистику результатов своей работы в файл. С помощью внешних средств, например утилиты `dig`, можно проверить, правильно ли функционирует механизм поиска имен.

Журнальная регистрация

Возможности подсистемы журнальной регистрации демона `named` действительно впечатляют. Сервер BIND использует систему Syslog для записи сообщений об ошиб-

[23] Такое поведение, очевидно, является непреднамеренным. Подробно об этом можно прочитать в официальном документе Microsoft DNS, на который есть ссылка в статье 286834 (`support.microsoft.com`) в базе знаний Microsoft.

ках и аномалиях. В последних версиях концепция журнальной регистрации обобщена: добавлен еще один уровень переадресации и поддерживается направление сообщений непосредственно в файлы. Прежде чем переходить к деталям, приведем мини-словарь терминов, связанных с журнальной регистрацией в пакете BIND (табл. 15.12).

📖 Подробнее о системе Syslog рассказывалось в главе 10.

Таблица 15.12. Лексикон пакета BIND

Термин	Что означает
Канал	Место, куда направляются сообщения: система Syslog, файл или устройство `/dev/null`[a]
Категория	Класс сообщений, генерируемых демоном `named`, например сообщения о динамических обновлениях или сообщения об ответах на запросы
Модуль	Имя исходного модуля, который сгенерировал сообщение
Средство	Название средства системы Syslog; за DNS не закреплено собственное средство, но можно выбрать любое стандартное
Важность	Степень важности сообщения об ошибке

[a] `/dev/null` — это псевдоустройство, отбрасывающее все входящие данные

Подсистема журнальной регистрации конфигурируется при помощи инструкции `logging` в файле `named.conf`. Сначала определяются каналы — возможные пункты доставки сообщений. Затем для различных категорий сообщений задаются каналы, куда они будут поступать.

При формировании сообщения ему назначается категория, модуль и уровень важности. После этого оно рассылается по всем связанным с категорией и модулем каналам. В каждом канале имеется фильтр, определяющий, сообщения какого уровня важности можно пропускать. Каналы, ведущие в систему Syslog, подвергаются дополнительной фильтрации в соответствии с правилами, установленными в файле `/etc/syslog.conf`.

Общий вид инструкции `logging` таков:

```
logging {
    определение_канала;
    определение_канала;
    ...
    category имя_категории {
        имя_канала;
        имя_канала;
        ...
    };
};
```

Каналы

Определение канала выглядит по-разному в зависимости от того, ведет ли он к файлу или к системе Syslog. Для каждого канала необходимо выбрать либо тип `file`, либо тип `syslog`; совмещать их канал не может.

```
channel имя_канала {

    file путь [versions число_версий | unlimited] [size размер];
    syslog средство;
    severity важность;
```

```
    print-category yes | no;
    print-severity yes | no;
    print-time yes | no;
};
```

Для файлового канала аргумент *число_версий* сообщает о том, сколько резервных копий файла нужно хранить. Аргумент *размер* задает предельно допустимый размер файла (примеры: `2048`, `100k`, `20m`, `15g`, `unlimited`, `default`), по достижении которого произойдет автоматическая ротация. К примеру, если файловый канал называется `mylog`, то в процессе ротации появятся версии `mylog.0`, `mylog.1` и т.д.

Для каналов системы Syslog задается имя средства, указываемое при регистрации сообщений. Это может быть любое стандартное средство, но на практике единственным разумным выбором являются средства `daemon` и `local0`—`local7`.

📖 Список средств системы Syslog приводился в разделе 10.4.

Остальные предложения в определении канала являются необязательными. Аргумент *важность* может принимать следующие значения (в порядке уменьшения приоритета): `critical`, `error`, `warning`, `notice`, `info` и `debug` (здесь может добавляться номер уровня, например `severity debug 3`). Понимается также значение `dynamic`, которое соответствует текущему уровню отладки сервера.

Параметры `print-*` задают или отменяют вывод различных префиксов сообщений. Система Syslog вставляет перед каждым сообщением метку времени, а также имя хоста, но не уровень важности или категорию. Существует также параметр, позволяющий отображать имя исходного файла (модуля), сгенерировавшего сообщение. Параметр `print-time` рекомендуется включать только для файловых каналов, поскольку в Syslog произойдет ненужное дублирование информации.

В табл. 15.13 перечислены четыре канала, которые определены по умолчанию. В большинстве случаев создавать новые каналы не требуется.

Таблица 15.13. Стандартные каналы журнальной регистрации пакета BIND

Имя канала	Назначение
`default_syslog`	Направляет сообщения уровня `info` и выше в систему Syslog от имени средства `daemon`
`default_debug`	Направляет сообщения в файл `named.run`, уровень важности устанавливается равным `dynamic`
`default_stderr`	Направляет сообщения в стандартный канал ошибок демона `named` с уровнем важности `info`
`null`	Отбрасывает все сообщения

Категории

Категории определяются программистом при написании кода; они распределяют зарегистрированные сообщения по теме или по выполняемым функциям, а не только по важности. В табл. 15.14 приводится текущий список категорий сообщений.

Журнальные сообщения

По умолчанию конфигурация регистрации сообщений выглядит следующим образом:

```
logging {
        category default { default_syslog; default_debug; };
};
```

После внесения существенных изменений в систему BIND необходимо просматривать журнальные файлы; возможно, стоит также повышать уровень отладки. После того как демон `named` вернется в стабильное состояние, настройте систему так, чтобы регистрировались только важные сообщения.

Журнал запросов — источник весьма полезной информации. На его основании можно проверить, работают ли директивы `allow`, узнать, кто посылает вам запросы, идентифицировать неправильно сконфигурированных клиентов и т.д. Регистрация запросов — хороший способ проверки сервера после крупномасштабных изменений его конфигурации, особенно если известна статистика запросов до внесения изменений.

Чтобы включить режим регистрации запросов, назначьте категории `queries` какой-нибудь канал. Регистрация в системе Syslog менее эффективна, чем непосредственно в файле, поэтому при регистрации каждого запроса создайте файловый канал, связанный с локальным диском. Зарезервируйте для журнала достаточный объем дискового пространства и будьте готовы отключить регистрацию, когда накопится достаточный объем данных. (Команда `rndc querylog` включает и отключает регистрацию запросов.)

Таблица 15.14. Категории сообщений пакета BIND

Категория	Что охватывает
`client`	Клиентские запросы
`config`	Ошибки анализа и обработки конфигурационного файла
`database`	Сообщения об операциях с базой данных
`default`	Категории, для которых не был явно назначен канал
`delegation-only`	Запросы передаются в NXDOMAIN зонами делегирования
`dispatch`	Отправка поступающих пакетов модулям сервера
`dnssec`	Сообщения протокола DNSSEC
`general`	Неклассифицированные сообщения
`lame-servers`	Сообщения о серверах, которые, как предполагается, обслуживают зону, но на самом деле это не так[a]
`network`	Сетевые операции
`notify`	Сообщения об изменениях зон
`queries`	Короткое сообщение для каждого (!) запроса, принимаемого сервером
`resolver`	Операции преобразования имен, например рекурсивный поиск, выполняемый от имени клиента
`security`	Принятые или не принятые запросы
`unmatched`	Запросы к `named` нельзя классифицировать (неподходящий класс, нет представления)
`update`	Сообщения о динамических обновлениях
`xfer-in`	Сообщения о зонных пересылках, принимаемых сервером
`xfer-out`	Сообщения о зонных пересылках, отправляемых сервером

[a] Это может быть как родительская, так и дочерняя зона; без исследования ничего нельзя сказать.

Отладка представлений может надоесть, но, к счастью, представление, которое совпадает с определенным запросом, регистрируется так же, как и сам запрос.

Ниже перечислены наиболее часто регистрируемые сообщения.

- *Неправильно сконфигурированный сервер* ("lame server"). Если подобное сообщение поступает от одной из внутренних зон, значит, в конфигурации системы имеется ошибка. Если же в сообщении говорится о какой-то зоне в Internet, то можно не беспокоиться: это "чужая" ошибка. Сообщения второго вида вполне можно отбрасывать, направляя их в канал `null`.
- *Неправильная отсылка* ("bad referral"). Это сообщение свидетельствует о неправильном взаимодействии серверов имен зоны.
- *Неавторитетен* ("not authoritative for"). Подчиненный сервер не может получить авторитетные данные о зоне. Возможно, у него хранится неправильный адрес главного сервера или же тот не смог загрузить зону.
- *Зона отклонена* ("zone rejected"). Демон `named` отказался загружать зонный файл, поскольку тот содержит ошибки.
- *Не задано стандартное значение TTL* ("no default TTL set"). Желательно задавать стандартное значение TTL посредством директивы `$TTL`, располагаемой в начале зонного файла. Ошибка свидетельствует о том, что такая директива отсутствует; в BIND 9 она нужна.
- *Нет корневых серверов имен для класса* ("no root nameservers for class"). Демону `named` не удается найти корневые серверы имен. Следует проверить файл корневых "подсказок" и факт подключения сервера к Internet.
- *Адрес уже используется* ("address already in use"). Порт, который нужен для работы демона `named`, занят другим процессом, возможно, другой копией демона. Если демон отсутствует в списке выполняющихся процессов, то, очевидно, он перед этим аварийно завершил свою работу и оставил открытым управляющий сокет утилиты `rndc`, который нужно найти и удалить[24]. Хороший способ устранить проблему — остановить и перезапустить процесс `named`:

```
% sudo /etc/init.d/named stop (или /sbin/service named stop в Red Hat)
% sudo /etc/init.d/named start (или /sbin/service named start в Red Hat)
```

- *Удален пакет с нулевым исходным портом* ("dropping source port zero packet from..."). В последних версиях пакета BIND разрешается задавать номер исходного порта запроса, и системные администраторы пользуются этим для добавления правил фильтрации DNS-пакетов на брандмауэрах. Однако следует помнить о том, что значение 0 не может быть номером порта TCP/UDP. Если сообщение касается одного из ваших хостов, исправьте соответствующий аргумент директивы `query-source` в файле `named.conf` этого хоста.
- *Обновление запрещено* ("denied update from..."). Имела место попытка динамического обновления зоны, которая была отвергнута вследствие установки `allow-update` или `update-policy` для зоны в файле `named.conf`.

[24] На организационном сервере верхнего уровня это сообщение может означать, что зона "com" все еще загружается. Ждите...

Пример конфигурации регистрации в журнале

Следующий фрагмент, взятый из файла ISC `named.conf` для интенсивно используемого сервера имен TLD, иллюстрирует все аспекты журнальной регистрации.

```
logging {
    channel default_log {
       file "log/named.log" versions 3 size 10m;
       print-time yes;
       print-category yes;
       print-severity yes;
       severity info;
};
category default { default_log; default_debug; };
channel xfer-log {
   file "log/xfer.log" versions 3 size 10m;
   print-category yes;
   print-severity yes;
   print-time yes;
   severity info;
};
   channel db-log {
      file "log/db.log" versions 3 size 1M;
      severity debug 1;
      print-severity yes;
      print-time yes;
   };
    category database { db-log; };
    category dnssec { xfer-log; };
    category xfer-in { xfer-log; };
    category xfer-out { xfer-log; };
    category notify { xfer-log; };
};
```

Уровни отладки

Уровни отладки демона `named` обозначаются целыми числами от 0 до 11. Чем выше число, тем больше текста содержит выходная информация. Уровень 0 выключает отладку. Уровни 1 и 2 отлично подходят для отладки конфигурационного файла и базы данных. Уровни выше четвертого предназначены для разработчиков кода демона.

Отладку можно включить из командной строки, запуская демон `named` с флагом `-d`. Например, команда

```
# named -d2
```

запускает демон на уровне отладки 2. По умолчанию отладочная информация записывается в файл `named.run`, находящийся в каталоге запуска демона. Этот файл растет очень быстро, поэтому во время отладки будьте внимательны.

Отладку можно также включать в процессе работы демона `named` с помощью команды `rndc trace`, которая увеличивает уровень отладки на единицу, или с помощью команды `rndc trace` *`уровень`*, которая задает уровню отладки указанное значение.

Команда `rndc notrace` выключает режим отладки. Кроме того, можно создать канал регистрации сообщений, в определение которого входит строка следующего вида:

```
severity debug 3
```

Она обеспечивает направление в канал всех отладочных сообщений вплоть до уровня 3. Другие директивы в определении канала указывают на то, куда в конечном итоге попадут сообщения. Чем выше уровень важности, тем больше информации регистрируется.

Просматривая журнальные файлы и отладочные сообщения, можно заметить, как часто делаются ошибки в конфигурации DNS. Одна-единственная точка в конце имени (точнее, ее отсутствие) приводит к угрожающему росту трафика DNS. Помните: точка нужна в конце каждого полностью определенного имени домена.

Отладка посредством команды `rndc`

В табл. 15.15 перечислены инструкции, которые принимает `rndc`. Если команду `rndc` выполнить без каких-либо аргументов, в результате будет получен список доступных команд. Некоторые инструкции приводят к созданию файлов. Такие файлы помещаются в каталог, обозначенный в файле `named.conf` как начальный каталог демона `named`.

Таблица 15.15. Отладочные инструкции команд `rndc`[а]

Инструкция	Функция
`dumpdb`	Запись образа базы данных DNS в файл `named_dump.db`
`flush` [*представление*]	Очистка всех кэшей или тех, которые относятся к указанному *представлению*
`flushname` *имя* [*представление*]	Очистка указанного *имени* от кэша сервера
`freeze` *зона* [*класс* [*представление*]]	Приостанавливает обновления в динамической *зоне*
`thaw` *зона* [*класс* [*представление*]]	Возобновляет обновления в динамической *зоне*
`halt`	Останов демона `named` без обработки отложенных обновлений
`querylog`	Переключение режима трассировки входящих запросов
`notrace`	Отключение отладки
`reconfig`	Повторная загрузка конфигурационного файла и всех новых зон
`refresh` *зона* [*класс* [*представление*]]	Ставит в очередь обслуживание *зоны*
`reload`	Повторная загрузка файла `named.conf` и зонных файлов
`reload` зона [*класс* [*представление*]]	Повторная загрузка только указанной зоны
`restart`[б]	Перезапускает сервер
`retransfer` *зона* [*класс* [*представление*]]	Повторно копирует данные для *зоны* из главного сервера
`stats`	Запись статистической информации в файл `named.stats`
`status`	Отображение текущего статуса выполнения демона `named`
`stop`	Останов демона `named`, а также (в BIND 9) сохранение отложенных обновлений
`trace`	Увеличение уровня отладки на единицу
`trace` *уровень*	Изменение уровня отладки

[а] Аргумент *класс* здесь обозначает то же самое, что и для записей о ресурсах; обычно это IN для Internet.

[б] Пока что не реализована в BIND 9 (9.3.0), хотя это обещают сделать.

Команда `rndc reload` заставляет демон `named` заново прочитать конфигурационный файл и повторно загрузить зонные файлы. Команду `rndc reload` *`зона`* удобно применять на интенсивно эксплуатируемом сервере, когда изменению подвергается только одна зона, а остальные трогать не нужно. Можно также указать класс и представление, чтобы перезагружать только выбранное представление данных зоны. Команда `rndc reconfig` упрощает добавление новых зон без перезагрузки; она повторно читает конфигурационный файл и загружает любые новые зоны, не нарушая существующие.

Команда `rndc freeze` *`зона`* останавливает динамические обновления и согласовывает журнальный файл незаконченных обновлений с файлами данных. После замораживания зоны вы можете редактировать зонные данные вручную. Пока зона будет оставаться замороженной, динамические обновления производиться не будут. После завершения редактирования воспользуйтесь командой `rndc thaw` *`зона`*, чтобы снова начать принимать динамические обновления.

Команда `rndc dumpdb` заставляет демон `named` записать образ своей базы данных в файл `named_dump.db`. Этот файл очень большой и содержит не только локальные данные, но также данные, накопленные в кэше сервера имен. Не так давно мы провели эту операцию на основном сервере имен, и вышло, что кэш базы данных занял 16 Мбайт, тогда как весь зонный файл — менее 200 Кбайт.

Если демон работает в среде с измененным корневым каталогом, то в команде `rndc` придется указывать путь к именованному сокету `rndc`, поскольку будет находиться не в каталоге `/var/run`, где его ожидает найти команда `rndc`. Это выглядит примерно так:

```
# sudo rndc -l /var/named/var/run/rndc
```

В BIND 9 важно убедиться, что версия демона `named` и версия команды `rndc` совпадают, иначе будет выдано сообщение о несовпадении версий протокола. Просто инсталлируйте их при следующей модернизации.

Сбор статистических данных в BIND

В BIND 9 не накапливается столь исчерпывающая статистическая информация, как в BIND 8. Вместо этого больше внимания уделяется итоговым данным. Получив соответствующее указание от команды `rndc`, демон `named` записывает имеющиеся статистические данные в файл `named.stats` в рабочем каталоге демона:

```
% sudo rndc stats
```

Вот пример полученных данных:

```
+++ Statistics Dump +++ (1088897026)
success    19917007
referral   981446459
nxrrset    9958824
nxdomain   199644113
recursion  0
failure    322556
--- Statistics Dump --- (1088897026)
```

Здесь показано число удачных операций преобразования имен и подытоживаются различные виды ошибок. Сервер успешно ответил на 19,9 миллионов запросов, произвел 980 миллионов отсылок, около 200 миллионов раз ответил “нет такого домена” (“no such domain”, `nxdomain`), примерно 10 миллионов раз ответил “нет такого набора записей о ресурсах” (“no such resource record set”, `nxrrset`) и не смог ответить 320 тысяч раз.

Было бы слишком наивно думать, будто источником 200 миллионов ошибок NXDOMAIN (около 20% от общей суммы, составляющей 1 миллиард запросов) стали опечатки и описки пользователей. В данном случае большинство ошибок было вызвано неверной конфигурацией Windows, а дефект в распознавателе Microsoft[25], который посылает много запросов к записям A, указывая в поле данных IP-адрес, а не доменное имя, как того требует спецификация DNS. Эти нелепые запросы адресованы явно фиктивному домену верхнего уровня ("56", "1" или все, что может стоять в последнем байте IP-адреса), поэтому они заставляют локальный сервер DNS обращаться к корневым серверам по поводу несуществующего домена. Отсюда и возникают ошибки NXDOMAIN.

Показатель `failure` отражает число ошибок, которые нельзя отнести ни к классу `nxdomain` (нет такого домена), ни к классу `nxrrset` (нет такого набора записей).

Отладка с помощью команды `dig`

Команды `nslookup`, `dig` и `host` позволяют в режиме командной строки посылать запросы базе данных DNS. Все эти команды входят в дистрибутив BIND. Самая старая из них — `nslookup`, она всегда являлась частью дистрибутива. Утилита `dig` (domain information groper — искатель доменной информации) первоначально была написана Стивом Хотцом (Steve Hotz), а Майкл Сойер (Michael Sawyer) переписал ее для BIND 9. Другая открытая утилита `host`, написанная Эриком Вассенааром (Eric Wassenaar), отображает информацию в удобном для восприятия виде и имеет функции для проверки синтаксиса зонных файлов.

Мы рекомендуем пользоваться командой `dig`, а не `nslookup`; команда `host` нам тоже нравится. Здесь мы рассмотрим детально только `dig`. Иногда вы можете получать разные результаты из-за того, что будете пользоваться разными модулями распознавания: в `dig` и `host` это распознаватель пакета BIND, а в `nslookup` — свой собственный.

По умолчанию `dig` использует локальный сервер имен и возвращает адресную информацию. Аргумент `@`*`сервер_имен`* организовывает запрос к определенному серверу имен[26]. Способность посылать запрос определенному серверу дает вам возможность проверить, дошли ли любые произведенные вами изменения в зоне до вторичных серверов и во внешний мир. Эта особенность вам очень пригодится, если вы будете использовать представления, и вам нужно будет проверить, сконфигурированы ли они корректным образом.

Если определить тип записи, `dig` запросит только этот тип. Псевдотип `ANY` возвращает все данные, связанные с определенным именем. Другим полезным флагом является `-x`: он меняет порядок байтов в IP-адресе на противоположный и делает за вас обратный запрос. Флаг `+trace` показывает итеративные шаги в процессе распознавания.

Результат работы команды `dig` выглядит примерно так:

```
$ dig yahoo.com

; <<>> DiG 9.3.0rc2 <<>> yahoo.com
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 16507
```

[25] Компания Microsoft "любезно" исправила эту ошибку в пакете SP2 к Win2K.

[26] Если сервер не существует, старые версии `dig` тихонько вернутся к локальному серверу, поэтому нужно внимательно следить за выходными данными. Начиная с версии BIND 9.2, в этой ситуации `dig` возвращает ошибку и не отвечает.

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 5, ADDITIONAL: 5

;; QUESTION SECTION:
;yahoo.com.           IN    A

;; ANSWER SECTION:
yahoo.com.       300    IN    A   66.94.234.13
yahoo.com.       300    IN    A   216.109.127.28
yahoo.com.       300    IN    A   216.109.127.29

;; AUTHORITY SECTION:
yahoo.com.       74804  IN    NS ns3.yahoo.com.
yahoo.com.       74804  IN    NS ns4.yahoo.com.
yahoo.com.       74804  IN    NS ns1.yahoo.com.
yahoo.com.       74804  IN    NS ns2.yahoo.com.

;; ADDITIONAL SECTION:
ns1.yahoo.com.    72593  IN    A  66.218.71.63
ns2.yahoo.com.    72596  IN    A  66.163.169.170
ns3.yahoo.com.    72594  IN    A  217.12.4.104
ns4.yahoo.com.    72594  IN    A  63.250.206.138

;; Query time: 6 msec
;; SERVER: 204.152.184.109#53(204.152.184.109)
;; WHEN: Sun Jul 9 16:03:43 2006
;; MSG SIZE rcvd: 245
```

`dig` включает отметки `aa` в списке `flags`, если ответ является авторитетным. Здесь мы спрашивали у локального сервера имен о домене “yahoo.com”, поэтому мы получили неавторитетный ответ.

Выходная информация команды `dig` включает не только сведения о домене, но и данные о количестве посланных запросов и полном времени получения ответа. Выводные данные отформатированы так, чтобы их можно было использовать в зонных файлах. Это особенно удобно, если делаются запросы к корневым серверам с целью заполнения файла “подсказок”. Точка с запятой в символе комментария в зонном файле используется командой `dig` для вставки комментариев в выходные данные.

Тестируя новую конфигурацию, не забудьте проверить как локальные, так и удаленные хосты. Если доступ к хосту можно получить по IP-адресу, но не по имени, то виновницей, скорее всего, является DNS.

Напрасное делегирование

Подавая заявку на получение доменного имени, вы просите, чтобы основному серверу имен и администратору DNS была выделена (делегирована) часть дерева имен. Если не поддерживать работу домена или изменить адреса серверов имен, не обновив связующие записи родительского домена, будет иметь место так называемое *напрасное делегирование* (lame delegation).

Последствия напрасного делегирования могут оказаться весьма негативными. Если пользователь попытается связаться с хостом, находящимся в неправильно сконфигурированном домене, сервер имен отклонит этот запрос, но клиент DNS продолжит повторять запрос сотни раз, терроризируя и главный сервер родительского домена, и корневые серверы.

Существует два способа обнаружить напрасное делегирование: пересматривая журнальные файлы или используя инструмент doc (domain obscenity control — контроль за "чистотой" домена). С примерами doc вы ознакомитесь в следующем разделе, а пока что давайте просмотрим некоторые записи журнала.

Многие сайты направляют канал регистрации lame-servers в каталог /dev/null и не беспокоятся о напрасном делегировании других людей. Эта мера будет оправдывать себя до тех пор, пока ваш собственный домен будет чистым, и не будет являться источником напрасного делегирования.

В одном журнальном файле, размер которого за неделю вырос до 3,5 Мбайт (на уровне info), более трети записей касались напрасного делегирования. Из них 16% запросов адресовались корневым серверам, предположительно по поводу несуществующих доменов. Один настойчивый пользователь более сотни раз запрашивал корневые серверы о домене tokyotopless.net. Бедняга... Ниже приведен пример журнального сообщения о напрасном делегировании:

```
Jan 29 05:34:52 ipn.caida.org named[223]: Lame server on
     'www.games.net' (in 'GAMES.net'?): [207.82.198.150].53
     'NS2.EXODUS.net'
```

Вот как можно установить источник проблемы с помощью команды dig (часть выходных данных отброшена):

```
% dig www.games.net.
;; ...
;; QUESTIONS:
;;    www.games.net, type = A, class = IN
;; ANSWERS:
www.games.net.      3600   A   209.1.23.92
;; AUTHORITY RECORDS:
games.net.          3600   NS  ns.exodus.net.
games.net.          3600   NS  ns2.exodus.net.
games.net.          3600   NS  ns.pcworld.com.
;; ADDITIONAL RECORDS: ...
```

В ответ на первый запрос к локальному серверу возвращается адресная запись для домена www.games.net и список авторитетных серверов. Сервер ns.exodus.net на момент обращения к нему работал нормально (результаты запроса здесь не показаны), а вот о сервере ns2.exodus.net этого сказать нельзя:

```
% dig @ns2.exodus.net www.games.net.
;; QUESTIONS:
;;    www.games.net, type = A, class = IN
;; AUTHORITY RECORDS:
net.               244362 NS  F.GTLD-SERVERS.net.
net.               244362 NS  J.GTLD-SERVERS.net.
net.               244362 NS  K.GTLD-SERVERS.net.
net.               244362 NS  A.GTLD-SERVERS.net.
;; ...
```

Сервер указан как авторитетный для домена, но для него не возвращаются записи, а лишь ссылки на серверы домена верхнего уровня "net". Следовательно, можно сделать вывод о том, что домен ns2.exodus.net сконфигурирован неправильно. Иногда при попытке обнаружить случаи напрасного делегирования на авторитетном сервере команда dig ничего не возвращает. В этом случае попробуйте запустить ее с флагом +norecurse, чтобы узнать, какая именно информация известна опрашиваемому серверу.

doc: контроль за "чистотой" домена

doc не является частью BIND. Сейчас за ним следит Бред Ноулес (Brad Knowles), и как раз на его сайте можно загрузить это инструмент:

```
www.shub-internet.org/brad/dns/ (Примечание: "shub", а не "shrub" (кустарник))
```

doc представляет собой сценарий оболочки, написанный на языке C. Если вы планируете поместить его в переменную path или запускать из cron, вы должны отредактировать сценарий и задать переменную auxd, чтобы она указывала на каталог для инсталляции.

doc проверяет делегирование, производя повторяющиеся обращения к dig. Он сообщает о несогласованностях, ошибках и прочих проблемах, связанных с определенным доменным именем. Его выходные данные содержат итоговые сведения обо всех обнаруженных проблемах; кроме того, он генерирует подробный журнальный файл в текущем каталоге.

Давайте посмотрим на некоторые примеры. Сначала мы посмотрим на artust.com, в котором, кажется, нет никаких проблем:

```
$ doc atrust.com
Doc-2.2.3: doc atrust.com
Doc-2.2.3: Starting test of atrust.com. parent is com.
Doc-2.2.3: Test date - Thu Jul 6 08:54:38 MDT 2006
Summary:
  No errors or warnings issued for atrust.com.
Done testing atrust.com. Thu Jul 6 08:54:46 MDT 2006
```

В журнальном файле (log.artust.com) отмечено, что "com" и "artust.com" согласны с идентичностями серверов имен artust.com, и что эти два сервера имеют один и тот же порядковый номер в своих записях SOA. doc проверяет также записи PTR для каждого сервера имен. Сами запросы dig, которые были выполнены инструментом doc, перечислены в конце журнального файла, вместе с откликами.

Наш следующий пример — домен cs.colorado.edu:

```
$ doc cs.colorado.edu
Doc-2.2.3: doc cs.colorado.edu
Doc-2.2.3: Starting test of cs.colorado.edu. parent is colorado.edu.
Doc-2.2.3: Test date - Thu Jul 6 08:55:15 MDT 2006
dig: Couldn't find server 'rs0.netsol.com.': No address associated with
hostname
DIGERR (NOT_ZONE): dig @rs0.netsol.com. for SOA of parent (colorado.edu.)
failed
DIGERR (NOT_AUTHORIZED): dig @pacifier.com. for SOA of cs.colorado.edu.
failed
dig: Couldn't find server 'xor.com.': No address associated with hostname
DIGERR (FORMAT_ERROR): dig @xor.com. for SOA of cs.colorado.edu. failed
Summary:
  ERRORS found for cs.colorado.edu. (count: 2)
  Incomplete test for cs.colorado.edu. (2)
Done testing cs.colorado.edu. Thu Jul 6 08:55:26 MDT 2006
```

Здесь мы видим, что в родительском домене, colorado.edu, есть несколько проблем. В зонных данных родительского домена неправильно записано, что rs0.netsol.com является сервером colorado.edu, а xor.com и pacifier.com являются серверами cs.colorado.edu. Чтобы посмотреть всю историю, нам нужно будет выполнить doc в

отношении родительского домена, colorado.edu, и просмотреть подробный журнальный файл.

И, наконец, давайте посмотрим на nsa.cctldmd.net:

```
$ doc nsa.cctldmd.net
Doc-2.2.3: doc nsa.cctldmd.net
Doc-2.2.3: Starting test of nsa.cctldmd.net. parent is cctldmd.net.
Doc-2.2.3: Test date - Thu Jul 6 08:56:20 MDT 2006
SYSerr: No servers for nsa.cctldmd.net. returned SOAs ...
Summary:
 YIKES: doc aborted while testing nsa.cctldmd.net. parent cctldmd.net.
 WARNINGS issued for nsa.cctldmd.net. (count: 2)
 Incomplete test for nsa.cctldmd.net. (1)
Done testing nsa.cctldmd.net. Thu Jul 6 08:56:22 MDT 2006
```

С домена cctld.net производится напрасное делегирование домену nsa.cctldmd.net. Следующая запись, взятая из файла log.nsa.cctldmd.net, проясняет эту проблему:

```
WARNING: register.dns.md. claims to be authoritative for nsa.cctldmd.net.
 == but no NS record at parent zone
```

Если вы управляете доменом, который включает поддомены (или не доверяете управляющим вашего родительского домена), выполняйте команду doc из cron каждую неделю, чтобы проверить корректность всех делегирований, имеющих отношение к вашему домену.

Другие средства контроля за "чистотой" DNS

Существует еще несколько других инструментов, проверяющих различные аспекты среды DNS. named-checkconf и named-checkzone поставляются вместе с BIND 9; они проверяют базовый синтаксис (но не семантику) файла named.conf и ваших зонных файлов. Самым первым инструментом проверки DNS был nslint, написанный Крейгом Лересом (Craig Leres) во время работы в лаборатории Lawrence Berkley Labs. Инструмент lamers (его можно загрузить с того же Web-сайта, что и doc) пересматривает журнальные файлы и посылает электронные письма DNS-администраторам сайтов-нарушителей, сообщая, что у них обнаружено напрасное делегирование и предлагая способ устранения проблемы. Инструмент DDT, созданный Джорджем Фразао (Jorge Frazao) и Артуром Ромао (Artur Romao), производит отладку кэшированных данных.

dnswalk просматривает дерево делегирований и выявляет несоответствия между родительскими и дочерними записями или между прямыми и обратными записями. Он также находит недостающие точки, ненужные записи и т.п. Он является базовым средством гигиены. Для полноты он должен также уметь выполнять зонные пересылки.

Проблемы, связанные с производительностью

В системе BIND 9 имеются некоторые проблемы с производительностью, которые стараются устранить в ISC. Ее производительность примерно равна производительности BIND 8, а эффективность в многопроцессорной архитектуре является очень низкой. Компьютеры f.root-servers.net используют BIND 9 и спокойно обрабатывают около 5000 запросов в секунду; такая производительность, пожалуй, будет приемлемой для большинства сайтов.

Мы уже неоднократно говорили, и повторим это еще раз: присваивайте параметру времени жизни разумные значения (недели или дни, но не часы или минуты). Использование короткого времени жизни будет помехой как для вас самих (из-за того, что вам придется постоянно повторно обслуживать одни и те же записи), так и для ваших Web-клиентов (из-за того, что клиентам постоянно придется выбирать их).

Страничная подкачка негативно сказывается на производительности сервера, причем проявляется это нелинейным образом. Следовательно, не стоит экономить на устройствах памяти серверов, на которых работает `named`. Вам придется ждать почти неделю, пока стабилизируется объем памяти, занятой демоном `named` (см. раздел 15.11). Используйте ретрансляторы. Архитектуру переадресации мы рассматривали в разделе 15.11.

15.17. Особенности пакета BIND в различных дистрибутивах

В этом разделе описаны особенности пакета в каждом из дистрибутивов и его стандартные настройки. Обновление дистрибутивов Linux будет происходить гладко, если новые версии пакета BIND были созданы в ISC.

Каждый из наших "справочных" дистрибутивов использует BIND и его сервер, предлагаемый по умолчанию, и только некоторые дистрибутивы включают этот пакет по умолчанию. Остальные дистрибутивы содержат утилиты BIND (`dig`, `nslookup`, `host` и т.д.), но не сами программы сервера имен. Вам нужно или указать, что вы хотите инсталлировать BIND при сборке нового компьютера или инсталлировать пакет BIND отдельно.

Клиент Linux BIND определяет свой домен, используемый по умолчанию, другие домены для поиска, когда имена не будут полностью определены, и IP-адреса его локальных серверов имен в каталоге `/etc/resolv.conf`. Linux-хост, работающий в качестве сервера BIND, использует файлы, перечисленные в табл. 15.16.

Таблица 15.16. Файлы пакета BIND в Linux

Файл	Каталог[a]	Описание
`resolv.conf`	`/etc`	Файл конфигурации распознавателя
`named, lwres`	`/usr/sbin`	Демон сервера имен
`named.conf`	`/etc`	Конфигурационный файл сервера имен в Red Hat и SuSE
`named.conf`	`/etc/bind`	Конфигурационный файл сервера имен в Debian, Ubuntu
`named-xfer`	`/usr/sbin`	Модуль зонных пересылок
`named.pid`	`/var/run/named`	Идентификатор процесса `named` (RHEL/Fedora)
`named.pid`	`/var/run/bind/run`	Идентификатор процесса `named` (Debian, Ubuntu)
`named.pid`	*домашний_каталог*`/var/run/named`	Идентификатор процесса `named` (SUSE)
`named.run`	*домашний_каталог*	Отладочная информация
`named.stats`	*домашний_каталог* *домашний_каталог*`/var/log`	Статистическая информация Статистическая информация (SUSE)
`named_dump.db`	*домашний_каталог* *домашний_каталог*`/var/log`	Образ всей базы данных Образ всей базы данных (SUSE)

[a] Под *домашний_каталог* понимается каталог, указанный в `named.conf` как домашний для файлов BIND.

В Linux есть файл "переключения служб" (/etc/nsswitch.conf), который определяет порядок обработки адресных привязок, задавая приоритеты различных источников данных. При отсутствии этого файла система по умолчанию принимает следующую конфигурацию:

```
hosts: dns [!UNAVAIL=return] files
```

Предложение !UNAVAIL означает следующее: если сервер DNS доступен, но имя не найдено, дальнейший поиск необходимо прекратить, а не переходить к следующему источнику (в данном случае к файлу /etc/hosts). Если же сервер имен не запущен (например, на этапе начальной загрузки), то система обратится к файлу hosts.

В наших дистрибутивах используются разные начальные установки:

```
hosts: files dns
```

Сказать, какая из этих конфигураций наилучшая, нельзя. Все зависит от особенностей конкретной системы. Желательно как можно больше информации о компьютерах хранить в DNS, а не в базе данных NIS или обычных файлах, однако мы стараемся также сохранить способность производить возврат к статическому файлу hosts во время процесса загрузки, если в этом возникнет необходимость.

При инсталляции RHEL или Fedora пакет BIND (сейчас это bind-9.2.3-13.1386.rpm) помещает двоичные файлы в каталог /usr/bin, страницы руководства — в каталог /usr/share/man, добавляет пользователя и группу "named" и создает каталоги для зонных файлов. Пользователь "named" имеет доступ к файлам данных посредством прав групп.

Если вы ничего не изменяли в каталоге /etc/sysconfig/named, то конфигурационный файл named.conf будет помещен в каталог /etc, а зонные файлы — в каталог /var/named. Никаких файлов-образцов не прилагается, хотя в пакете bindconf они должны быть.

Debian и Ubuntu позволяют выбрать между BIND 8 и BIND 9, в зависимости от того, какой пакет вы инсталлируете (bind или bind9). Ниже даны подробности BIND 9.

Программы и файлы имеют "связку" владелец root/групповой владелец с правом на доступ, если демон named будет вызываться от имени этой "связки", а не от имени root. Некоторые полезные файлы-образцы помещаются в каталог /etc/bind. К ним относится файл named.conf и зонные файлы для корневых "подсказок", зоны localhost, широковещательных адресов и пространства приватных адресов. Прилагаемый файл named.conf включает файлы named.conf.options и named.conf.local. В качестве каталога по умолчанию для BIND выбран каталог /var/cache/bind; в момент поставки каталог существует, но является пустым. Логика информации о конфигурации, находящейся в каталоге /etc, и информации о зоне в каталоге /var состоит в том, что если вы работаете на сервере, который по отношению к другим сайтам является вторичным, то вам не нужно будет контролировать размер зонных файлов, которые будет записывать демон named, поэтому вам, скорее всего, придется хранить файлы в каталоге /var. Зоны, по отношению к которым вы работаете на основном сервере, могут либо существовать с конфигурационными файлами и использовать абсолютные пути в файле named.conf, либо находиться в каталоге /var/cache/bind. В Debian файл-образец named.conf не придется редактировать, если планируется запустить только кэширующий сервер.

Вы должны будете добавить зоны, в отношении которых ваш сервер является авторитетным, желательно в приготовленный файл `named.conf.local`.

Файлы-образцы в Debian используют некоторые новые особенности BIND, которые помогут вашим серверам быть "хорошими DNS-жителями" в сети. Например, они конфигурируют зоны "com" и "net" как зоны для делегирования (delegation-only), чтобы при опечатках, допускаемых пользователями, не генерировалась реклама посредством инструмента Site Finder. Если вы не используете приватное адресное пространство (документ RFC1918), то пустые зонные файлы RFC1918 предотвратят утечку этих адресов из локальной сети. Каталог `/usr/share/doc/bind9` содержит несколько полезных ссылок. В файле `README.Debian` (даже в Ubuntu) описана стратегия Debian в конфигурировании BIND.

Инсталлировать BIND из пакета `rpm` в SUSE было бесполезно по причине отсутствия зависимостей, но как только эта проблема была устранена, все стало на свои места. Во время инсталляции дается пояснение происходящего и предлагаются подробные сведения об инсталляции сервера имен. Мастер SUSE впоследствии стал использовать YaST вместо `rpm`, и, пожалуй, это был верный ход.

По умолчанию `named` работает в среде с измененным корневым каталогом внутри `/var/lib/named` в качестве пользователя и группы "named". Инсталлятор создает "каталог-темницу" `chroot` и помещает в него файлы, необходимые для работы демона `named`, и даже такие деликатные вещи как сокет домена UNIX для Syslog. Дополнительные конфигурационные файлы (отличные от файла `named.conf`) и зонные файлы находятся в каталоге `/etc/named.d` и копируются в `chroot` в начале работы демона `named`. Если вы не хотите, чтобы демон `named` запускался из измененного корневого каталога, измените строку

```
NAMED_RUN_CHROOTED="yes"
```

в `/etc/sysconfig/named`. Это единственное, что вам нужно будет изменить; сценарии запуска в каталоге `/etc/init.d` ссылаются на эту информацию и способны запускать `named` любым образом.

SUSE предлагает образец файла `/etc/named.conf`, содержащий полезные комментарии, которые объясняют многие параметры. Файл `/etc/named.conf` в SUSE несколько сложноват для восприятия. Файл, используемый по умолчанию, импортирует файл `named.conf.include`, который затем импортирует файл `rnd-access.conf` из `/etc/named.d`, каждый из которых нормально воспринимается пользователями. Не совсем понятно, что в SUSE имели в виду в плане безопасности. `rndc` предварительно сконфигурирована для принятия управляющих инструкций только из адреса `localhost`.

Файл `named.conf` в SUSE можно использовать как есть для запуска только кэширующего сервера. Если вам нужно обслуживать файлы вашей зоны, поместите зонные файлы в каталог `/etc/named.d` и перечислите имена зон в файле `/etc/named.conf.include`.

Документация ISC по системе BIND находится в каталоге `/usr/share/doc/packages/bind9`.

15.18. Рекомендуемая литература

Система доменных имен и пакет BIND описаны во многих источниках, включая документацию, входящую в состав пакета, отдельные главы в книгах об Internet, несколько книг серии "In a Nutshell" издательства O'Reilly, а также многочисленные онлайновые ресурсы.

Списки рассылки

Ниже перечислены списки рассылки, посвященные пакету BIND.

- `bind-announce` — чтобы подписаться, отправьте сообщение по адресу `bind-announce-request@isc.org`;
- `namedroppers` — чтобы подписаться, отправьте сообщение по адресу `namedroppers-request@internic.net`.
- `bind-users` — чтобы подписаться, отправьте сообщение по адресу `bind-users-request@isc.org`;
- `bind9-workers` — чтобы подписаться, отправьте сообщение по адресу `bind9-workers-request@isc-org` (для разработчиков пакета).

Сообщать о найденных ошибках можно по адресу `bind-bugs@isc.org` или `bind9-bugs@isc.org`.

Книги и другая документация

- The Nominum and ISC BIND Development Teams. *BINDv9 Administrator Reference Manual.*

  Документ включен в дистрибутив BIND (`doc/arm`), а также доступен на Web-сайте `www.isc.org`. Новая версия документации BIND, *BIND Reference Manual* (возможно, `doc/brm`) уже разработана в ISC, но пока что не доступна для изучения. Более ранний документ, *BIND Operations Guide*, или BOG, как его называют, содержит подробное описание процессов функционирования и конфигурирования пакета BIND 4. Документ BOG входит в дистрибутивы BIND вплоть до версии 8.

- Albitz, Paul, and Cricket Liu. *DNS and BIND, 4th Edition*. Sebastopol, CA: O'Reilly, 2001.

  Это популярная и очень уважаемая книга о пакете BIND, содержащая описание версий BIND 8 и BIND 9. Полнота изложения впечатляет.

Ресурсы в Internet

Каталог ресурсов DNS (`www.dnsresources.com`) — это полезная коллекция ресурсов и ссылок на них.

В поисковой системе Google индексированный список ресурсов DNS находится по следующему адресу:

```
http://directory.google.com/Top/Computers/Internet/Protocols/DNS
```

Полезную информацию можно также найти в конференции:

```
comp.protocols.dns.bind
```

Документы RFC

RFC-документы, в которых описывается система доменных имен, доступны на Web-сайте `www.rfc-editor.org`. Ниже перечислены избранные подмножества этих документов. Мы привыкли перечислять страницу или около того самой важной информации, связанной с DNS, однако сейчас их настолько много (более 100, причем кроме них еще есть полсотни черновиков в Internet), что вам лучше перейти на страницу rfc-editor.org, чтобы получить доступ ко всему архиву. В каталогах `doc/rfc` и `doc/draft` можно поискать сведения о текущем дистрибутиве BIND.

Ниже перечислены исходные спецификации, стандарты для DNS, начиная с 1987 г.

- 1034 — "Domain Names: Concepts and Facilities" (Доменные имена: концепции и базовые возможности);
- 1035 — "Domain Names: Implementation and Specification" (Доменные имена: реализация и спецификация).

15.19. Упражнения

15.1. Поясните назначение следующих записей о ресурсах: SOA, PTR, A, MX и CNAME.

15.2. Что такое связующие записи и зачем они нужны? Найдите с помощью команды `dig` записи, связывающие вашу локальную и родительскую зоны.

15.3. В чем смысл отрицательного кэширования? Почему оно так важно?

15.4. Создайте для вашего сайта псевдозаписи SPF, которые помогут бороться со спамом.

☆ **15.5.** Какие действия необходимо предпринять для создания домена второго уровня? Назовите технические и организационные мероприятия.

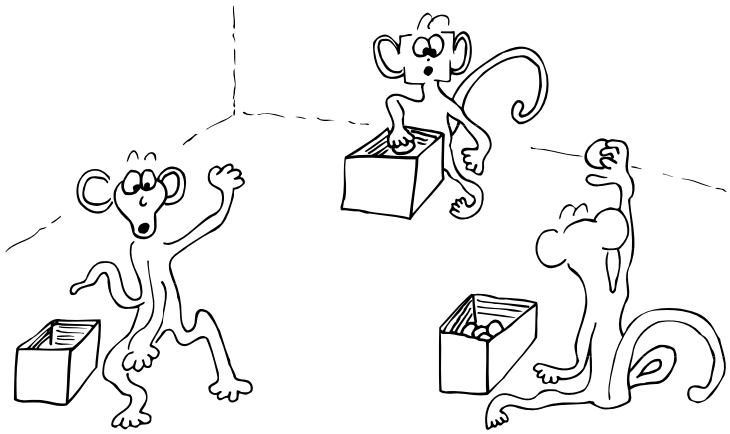
☆ **15.6.** В чем разница между авторитетным и неавторитетным ответом на DNS-запрос? Как убедиться в том, что ответ действительно авторитетен?

☆ **15.7.** Какой компьютер является вашим локальным сервером имен? Какие действия придется ему предпринимать для распознавания имени www.admin.com (предполагается, что ни в одном из кэшей DNS информации об этом домене нет)?

☆ **15.8.** Объясните, какое влияние на DNS оказывает 512-байтовое ограничение на размер пакета, налагаемое протоколом UDP? Какие существуют решения потенциальных проблем?

Глава 16

Сетевая файловая система



Сетевая файловая система, или NFS (Network File System), позволяет пользователям совместно работать с файлами, расположенными на разных компьютерах. NFS почти прозрачна для пользователей и не поддерживает понятие сеанса, т.е. при сбое сервера никакие данные в удаленных файловых системах не пропадают. Клиенты просто ждут, когда сервер вновь начнет функционировать, а затем продолжают работать так, будто ничего не произошло.

Сетевую файловую систему разработала компания Sun Microsystems в 1985 г. Первоначально NFS была реализована как суррогат файловой системы для бездисковых клиентов, однако предложенный протокол оказался столь удачным, что со временем стал универсальным решением проблемы совместного использования файлов. Сейчас уже трудно припомнить, какой была жизнь до появления NFS. Система NFS поддерживается практически во всех полнофункциональных дистрибутивах Linux.

16.1. Общая информация об NFS

В наши дни NFS используется только для совместного использования файлов в среде Linux и UNIX. Windows-клиентам для обслуживания файлов необходимо применять CIFS/Samba. Samba и CIFS будут рассмотрены в разделе 26.6.

NFS состоит из целого ряда компонентов, включая протокол и сервер монтирования, демоны, реализующие дистанционный доступ к файлам, а также несколько диагностических утилит. Части клиентского и серверного программного обеспечения NFS находятся непосредственно в ядре. К счастью, они не требуют конфигурирования и преимущественно “прозрачны” с точки зрения администратора.

Версии протокола NFS

Протокол NFS отличается удивительной стабильностью. Первая открытая версия NFS имела номер 2. В начале 90-х гг. в протокол был внесен ряд изменений, что привело к появлению версии 3. Ее характеризовали повышенная производительность и улучшенная поддержка больших файлов. Сейчас доступны первые варианты версии 4. Эта версия характеризуется множеством новых улучшений, о которых будет сказано ниже.

В NFS версии 2 клиент не мог считать операцию записи завершенной, не получив подтверждения от сервера. Поэтому сервер должен был записать модифицированный блок на диск, прежде чем посылать подтверждение. Это позволяло избежать проблем в случае сбоя сервера. Такое ограничение существенно замедляло запись, поскольку во многих случаях модифицированные блоки вполне можно было хранить лишь в резидентном буфере.

В версии 3 указанное узкое место было устранено за счет схемы согласования, позволяющей выполнять асинхронные записи. Были улучшены также другие аспекты протокола, связанные с производительностью, вследствие чего NFS 3 стала работать гораздо быстрее, чем NFS 2. Сервер версии 3 всегда может взаимодействовать с сервером версии 2. Просто он при этом переключается на использование более старого протокола.

NFS версии 4 все еще находится в разработке и пока что не поддерживается в Linux. Для нее нужно ядро 2.6.1 или выше; кроме того, ее необходимо вручную включать в ядре. К улучшениям относится следующее:

- совместимость и взаимодействие с брандмауэрами и устройствами NAT;
- внедрение протоколов блокировки и монтирования в основу протокола NFS;
- выполнение операций с сохранением информации о состоянии;
- более надежная и интегрированная защита;
- поддержка дублирования и миграции;
- поддержка UNIX- и Windows-клиентов;
- работа со списками управления доступом;
- поддержка имен файлов в формате Unicode;
- хорошая производительность даже при низкоскоростном соединении.

О текущем состоянии NFS версии 4 в Linux и ссылки на самые последние версии программного обеспечения можно узнать по адресу `nfs.sourceforge.net`.

Так как NFS версии 4 по-прежнему находится на стадии разработки, мы не будем обсуждать ее детально в этой главе. Тем не менее, о ней забывать не следует; многие из запланированных особенностей направлены на устранение давно существующих недостатков NFS. Будем надеяться, что нам повезет увидеть наиболее перспективную спецификацию версии 4.

Выбор транспортного протокола

В основе NFS лежит механизм RPC (Remote Procedure Call — дистанционный вызов процедур) компании Sun, который определяет системно-независимый способ взаимодействия процессов по сети. Положительным побочным эффектом такого решения является возможность выбора транспортного протокола: TCP или UDP.

Изначально в NFS применялся протокол UDP, поскольку он обеспечивал наилучшую производительность в локальных сетях 80-х гг. И хотя программное обеспечение

NFS само выполняет сборку пакетов и осуществляет контроль ошибок, но ни в UDP, ни в NFS не реализованы алгоритмы управления перегрузкой, которые необходимы для достижения нормальной производительности в крупных IP-сетях.

С целью решения этих проблем все современные системы теперь позволяют использовать в качестве транспортного протокола для NFS не UDP, а TCP. Первоначально эта возможность предназначалась для обеспечения работы NFS в сетях с маршрутизаторами и в Internet. Но в настоящее время большинство специалистов рассматривают TCP и как оптимальное средство для управления локальным NFS-трафиком. По мере удешевления памяти и роста производительности центральных процессоров и сетевых контроллеров первоначальные доводы в пользу UDP становятся все менее актуальными. В системе Linux поддержка TCP на сервере NFS стала возможной, начиная с версии 2.4 ядра.

Серверы, поддерживающие протокол TCP, как правило, принимают запросы на установление соединений от обоих транспортных протоколов, поэтому выбор между TCP и UDP делает клиент. Клиент выбирает предпочтительный для себя вариант с помощью параметра команды `mount` (если монтирование выполняется вручную) или в конфигурационном файле, таком как `/etc/fstab`.

Блокировка файлов

Блокировка файлов (реализуемая системными вызовами `flock`, `lockf` и/или `fcntl`) в течение долгого времени была слабым звеном в UNIX. В локальных файловых системах этот механизм был воплощен далеко не идеально. Что касается NFS, то положение все еще неустойчиво. Серверы NFS не поддерживают понятие сеанса: им не известно, какие компьютеры работают с конкретным файлом. Но эта информация необходима для использования блокировок. Как быть?

Традиционное решение заключается в реализации механизма файловых блокировок отдельно от NFS. В большинстве систем этой цели служат два демона: `lockd` и `statd`. К сожалению, по ряду причин они работают не оптимальным образом.

Дисковые квоты

Доступ к информации о дисковых квотах на удаленном компьютере осуществляет серверный демон `rquotad`, также работающий отдельно от NFS. Серверы NFS поддерживают дисковые квоты, но пользователи не смогут ничего о них узнать, если на удаленном компьютере не работает демон `rquotad`.

Мы считаем механизм дисковых квот морально устаревшим; тем не менее, в некоторых организациях с помощью дисковых квот по-прежнему вводятся ограничения на использование дискового пространства. Если вам приходится контактировать с одной из таких организаций, вы можете проконсультироваться на предмет квот в определенном дистрибутиве Linux. Больше мы не будем рассказывать об этом демоне.

Секретные ключи и монтирование без учета состояния

Прежде чем начать работать с файловой системой NFS, клиент должен ее смонтировать, как если бы это была файловая система, находящаяся на локальном диске. Но NFS не хранит информацию о сеансах, поэтому сервер не “знает”, какие клиенты смонтировали ту или иную файловую систему. Вместо этого сервер вручает клиенту секретный

ключ по факту успешного завершения операции монтирования. Этот ключ идентифицирует каталог монтирования на сервере NFS и дает клиенту возможность получить доступ к содержимому каталога.

Как правило, при размонтировании и повторном монтировании файловой системы на сервере ее секретный ключ меняется. Но есть особый случай: ключ сохраняется при перезагрузке, чтобы сервер после сбоя смог вернуться в предыдущее состояние. Не стоит, однако, пробовать загружаться в однопользовательском режиме, работать с файловой системой, а затем загружаться вновь: это приведет к аннулированию ключей и лишит клиентов доступа к файловым системам, которые они смонтировали. Пользователям придется либо перезагрузиться, либо осуществить повторное монтирование файловых систем.

Если у клиента есть секретный ключ, он может посредством механизма RPC посылать файловой системе запросы, например на создание файла или чтение блока данных. Поскольку в NFS нет понятия сеанса, клиент самостоятельно должен убедиться в том, что сервер подтвердил запрос на запись.

Соглашения об именах совместно используемых файловых систем

Управлять сетевыми файловыми системами проще, если ваша схема именования включает имя каждого удаленного сервера (например, `/anchor/tools` — это файловая система, расположенная на компьютере `anchor`). Такие имена очень полезны, потому что они позволяют переводить объявления вида “компьютер `anchor` будет всю субботу отключен в связи с модификацией” в полезную информацию “я не смогу поработать над файлом `/anchor/tools/TeX` в субботу, чтобы закончить диссертацию, поэтому лучше отправлюсь на лыжную прогулку”.

К сожалению, эта схема требует, чтобы каталог `/anchor` существовал в корневых каталогах всех клиентских компьютеров. Если клиент получает доступ к файловым системам на нескольких компьютерах, его корневой каталог может оказаться “захламленным”. Рассмотрите возможность углубления иерархии: например, используйте каталоги `/home/anchor`, `/home/rastadon` и т.д. Мы советуем прибегнуть к помощи демонов автоматического монтирования, описанных начиная с раздела 16.6.

Безопасность и NFS

NFS представляет собой удобное средство доступа к файлам по сети, потому является источником серьезным угроз для безопасности. NFS версии 3 можно использовать в качестве тренажера для проверки брешей, существующих или когда-либо существовавших в UNIX и Linux. Изначально в NFS не были предусмотрены никакие средства защиты — такова цена, которую пользователи вынуждены были платить за удобство работы. К счастью, в Linux имеются средства защиты, позволяющие решить многие традиционные проблемы NFS.

Доступ к томам NFS предоставляют записи файла `/etc/exports`. В нем перечислены сетевые имена (или IP-адреса) компьютеров, которым разрешен доступ к файловым системам сервера. Конечно, о должной степени безопасности говорить не приходится, так как сервер не проверяет подлинность клиентов. Подменить идентификаторы клиента не так уж сложно, поэтому доверять такому способу взаимодействия нельзя. Но меры предосторожности следует соблюдать в полном объеме: файловые системы должны экс-

портироваться только проверенным клиентам. Следите за тем, чтобы случайно не сделать файловую систему общедоступной.

🕮 Подробнее о файле `exports` рассказывается далее в разделе 16.2.

Доступ к портам NFS всегда должен быть строго ограничен. К счастью, во всех версиях Linux имеется брандмауэр, который может справиться с этой задачей.

Как и в локальных файловых системах, в NFS доступ к файлам контролируется посредством идентификаторов пользователей и групп, а также с помощью прав доступа. Но, опять-таки, сервер NFS доверяет сведениям, которые клиент предоставляет о себе. Если пользователи mary и bob имеют одинаковый идентификатор на двух разных клиентских компьютерах, они смогут получить доступ к файлам друг друга. Кроме того, если пользователь имеет доступ к учетной записи root, он сможет подменить любой идентификатор — и ничего не подозревающий сервер разрешит ему обратиться к соответствующим файлам. По этой причине мы настоятельно рекомендуем пользоваться глобально уникальными идентификаторами и описанной в табл. 16.3 опцией `root_squash`.

В хорошо известном нам крупном ВУЗе допустили ошибку: они не использовали эту опцию. Как следствие, 5 больших серверов и 60 настольных ПК были атакованы. На устранение последствий ушло несколько дней.

🕮 Подробнее о сетевом брандмауэре можно прочитать в разделе 20.12.

При наличии брандмауэра заблокируйте доступ к TCP- и UDP-портам 2049, которые используются протоколом NFS. Кроме того, необходимо перекрыть доступ к демону `portmap`, который обычно прослушивает TCP- и UDP-порты 111. За всеми этими предосторожностями не стоит также забывать правило, обычно подразумеваемое неявно: сетевые файловые системы не должны экспортироваться на нелокальные компьютеры или экспортироваться по открытому Internet-соединению.

Учетные записи `root` и `nobody`

В общем случае пользователи должны иметь одинаковые привилегии в любой системе, к которой они получают доступ. Однако традиционно стараются ограничивать возможности неконтролируемого применения прав суперпользователя в файловых системах, смонтированных посредством NFS. По умолчанию сервер NFS в Linux перехватывает входящие запросы, посылаемые от имени пользователя с идентификатором 0, и "делает вид", будто они поступают от другого пользователя. Таким образом, учетная запись `root` не отменяется, но уравнивается в правах с обычными учетными записями.

Для этого специально определена фиктивная учетная запись `nobody`, чтобы под нее "маскировался" пользователь root, работающий на сервере NFS. Традиционно ее идентификатор равен 65534 (обратный код числа -2)[1]. Изменить значения UID и GID для пользователя root позволяют опции экспортирования `anonuid` и `anongid`. С помощью опции `all_squash` можно заставить систему преобразовывать все клиентские идентификаторы пользователей в одинаковый серверный идентификатор. В такой конфигурации все различия между пользователями стерты и образуется нечто вроде файловой системы общего доступа.

[1] На сервере NFS в Red Hat стандартное значение UID равно -2, тогда как в файле `passwd` учетной записи `nobody` присвоен идентификатор 99. Можно оставить все как есть, добавить в файл `passwd` запись с идентификатором -2 или задать параметры `anonuid` и `anongid` равными 99. В других дистрибутивах учетные записи `nobody` и `nogroup` в файле `passwd` имеют привычный идентификатор -2.

В то же время существует и противоположное средство — опция `no_root_squash`, отключающая преобразование идентификатора пользователя root. Она иногда нужна для поддержки бездисковых клиентов и программ, которым необходим доступ к файловой системе с правами суперпользователя. Как правило, включать эту опцию не нужно, иначе пользователи, обладающие на стороне клиента привилегиями пользователя root, смогут модифицировать защищенные файлы. Несмотря на это, данная опция является доступной.

Все эти предосторожности преследуют благородную цель, однако конечный результат далек от идеала. Даже будучи клиентом NFS, пользователь root может с помощью команды `su` "принять облик" любого пользователя, так что файлы никогда не бывают полностью защищены. Системные учетные записи, такие как `bin` и `sys`, не подвергаются смене идентификатора, поэтому принадлежащие им файлы (некоторые системные программы и сторонние приложения) открыты для нападений. Единственный эффект от смены идентификаторов заключается в том, что предотвращается доступ к файлам, которые принадлежат пользователю root и недоступны для чтения или записи остальным пользователям.

16.2. Серверная часть NFS

Говорят, что сервер "экспортирует" файловую систему, если он делает ее доступной для использования другими компьютерами.

В NFS версии 3 процесс монтирования файловой системы (т.е. получения клиентом секретного ключа) отличается от процесса последующего доступа к файлам. В обеих операциях используются совершенно разные протоколы, и запросы обслуживаются разными демонами: в первом случае — это `mountd`, во втором — `nfsd`. На самом деле эти демоны называются соответственно `rpc.mountd` и `rpc.nfsd` как напоминание о том, что их работа основана на механизме RPC (следовательно, для их выполнения требуется демон `portmap`, о котором рассказывается в конце раздела 29.3). Но для простоты мы будем опускать префикс `rpc`.

На сервере NFS оба демона, `mountd` и `nfsd`, должны запускаться на этапе начальной загрузки системы и выполняться, пока система не будет выключена. Чаще всего вызов демонов осуществляется сценариями запуска системы автоматически, при условии, что экспортируемые файловые системы сконфигурированы. Имена сценариев запуска сервера NFS в разных дистрибутивах перечислены в табл. 16.1.

Таблица 16.1. Сценарии запуска сервера NFS

Дистрибутив	Сценарии
Red Hat	`/etc/init.d/nfs`
Fedora	`/etc/rc.d/init.d/nfs`
SUSE	`/etc/init.d/nfsserver`*
Debian	`/etc/init.d/nfs-kernel-server` `/etc/init.d/nfs-common`
Ubuntu	`/etc/init.d/nfs-kernel-server` `/etc/init.d/nfs-common`

* В SUSE сценарий `/etc/init.d/nfs` монтирует клиентские файловые системы NFS.

Демоны mountd и nfsd совместно эксплуатируют общую базу данных с информацией о правах доступа, где указано, какие файловые системы следует экспортировать и какие клиенты могут их монтировать. Рабочая копия этой базы данных обычно хранится в файле /usr/lib/nfs/xtab; часть таблиц находится в ядре. Поскольку файл базы данных представлен в нечитабельном формате, для добавления и редактирования записей существует вспомогательная утилита exportfs. Чтобы удалить записи, воспользуйтесь командой exportfs -u.

В большинстве систем имеется файл /etc/exports, содержащий список экспортируемых каталогов в текстовом виде. По умолчанию все файловые системы, указанные в этом файле, экспортируются на этапе начальной загрузки. С помощью команды exportfs -a можно самостоятельно выполнить экспорт после внесения изменений в файл exports. В командной строке утилиты exportfs можно указать имя клиента, путь к нужному каталогу и различные опции.

NFS работает не на физическом, а на логическом уровне файловой системы. Экспортировать можно любой каталог; не обязательно, чтобы он был точкой монтирования или корневым каталогом физической файловой системы. Тем не менее, из соображений безопасности NFS определяет границы между файловыми системами, относящимися к различным устройствам, и требует, чтобы каждая такая файловая система экспортировалась отдельно. К примеру, если раздел смонтирован как /users, то корневой каталог можно экспортировать независимо от этого раздела.

Обычно клиентам разрешается монтировать подкаталоги экспортированного каталога. Скажем, если сервер экспортирует каталог /chimchim/users, то клиент может смонтировать подкаталог /chimchim/users/joe, а остальную часть каталога users — проигнорировать. Большинство версий UNIX не позволяет экспортировать подкаталоги с параметрами, отличными от параметров родительского экспортируемого каталога, но Linux — исключение из правила.

Файл exports

В файле /etc/exports перечисляются файловые системы, экспортируемые посредством NFS, к которым могут получать доступ клиенты. Файловая система отделена от списка клиентов пробелом, а сразу после имени клиента в круглых скобках стоит список связанных с ним опций, разделенных запятыми. Длинные строки разрешается разбивать на части с помощью обратной косой черты.

Вот образцы записей:

```
/home/boggs          inura(rw,no_root_squash) lappie(rw)
/usr/share/man       *.toadranch.com(ro)
```

Нельзя задать список клиентов с одинаковым набором опций, хотя некоторые типы "клиентов" соответствуют многим узлам. В табл. 16.2 перечислены четыре типа клиентских спецификаций, которые могут встречаться в файле exports.

Таблица 16.2. Клиентские спецификации в файле /etc/exports

Тип	Синтаксис	Назначение
Имя хоста	*имя_хоста*	Имена отдельных хостов
Сетевая группа	*@имя_группы*	Сетевые группы NIS; подробнее об этом рассказывается в разделе 17.4
Подстановочные знаки	* и ?	Полностью определенные доменные имена с подстановочными знаками; метасимвол '*' не соответствует точке
IP-сети	*IP-адрес/маска*	Спецификации в формате CIDR (например, 128.138.92.128/25)

В табл. 16.3 описаны наиболее распространенные опции экспортирования файловых систем.

Таблица 16.3. Распространенные опции экспортирования

Опция	Описание
`ro`	Экспорт только для чтения
`rw`	Экспорт для чтения и записи (по умолчанию)
`rw=`*`список`*	Экспорт главным образом для чтения; в списке перечислены клиенты, которым разрешено монтирование для записи, в то время как все остальные клиенты должны монтировать файловую систему только для чтения
`root_squash`	Подмена значений UID и GID, равных 0, значениями, указанными в опциях `anonuid` и `anongid`; это установка по умолчанию
`no_root_squash`	Разрешает обычный доступ от имени пользователя root (небезопасно)
`all_squash`	Подмена идентификаторов всех пользователей и групп их анонимными версиями; это удобно при работе с персональными компьютерами и ненадежными однопользовательскими станциями
`anonuid=`*`xxx`*	Значение UID, которое нужно использовать для запросов, поступающих от пользователя root
`anongid=`*`xxx`*	Значение GID, которое нужно использовать для запросов, поступающих от пользователя root
`secure`	Требует, чтобы запрос на удаленный доступ поступал из привилегированного порта
`insecure`	Разрешает удаленный доступ с любого порта
`noaccess`	Предотвращает доступ к каталогу и его подкаталогам (используется при экспорте вложенных каталогов)
`wdelay`	Задержка записей в надежде на объединение множества обновлений
`no_wdelay`	Немедленная запись данных на диск
`async`	Заставляет сервер отвечать на запросы на запись, прежде чем выполнять сами операции записи на диск
`nohide`	Показывает файловые системы, смонтированные внутри экспортированных деревьев файлов
`hide`	Обратное действие `nohide`
`subtree_check`	Проверяет, находится ли каждый запрошенный файл внутри экспортированного поддерева
`no_subtree_check`	Проверяет, относятся ли запросы на данный файл к экспортированной файловой системе
`secure_locks`	Требует авторизовать все запросы на блокировку
`insecure_locks`	Определяет менее строгие критерии блокировки (поддерживает старые клиенты)
`auth_nlm`	Синоним `secure_locks`
`no_auth_nlm`	Синоним `insecure_locks`

Сервер NFS в Linux обладает необычной возможностью: он разрешает экспортировать родительский каталог с одними опциями, а подкаталоги — с другими. Имеется также опция noaccess, позволяющая отменить экспорт подкаталогов, к которым не нужно предоставлять удаленный доступ.

Например, в конфигурации

```
/home           *.toadranch.com(rw)
/home/boggs     (noaccess)
```

узлам домена toadranch.com разрешен доступ ко всем каталогам файловой системы /home, кроме /home/boggs. Отсутствие имени клиента во второй строке означает, что установка относится ко всем узлам. Так, очевидно, безопаснее.

Опция subtree_check (используется по умолчанию) проверяет, находятся ли файлы, к которым обращается клиент, внутри экспортированного подкаталога. Если эту опцию выключить, то проверяться будет только факт нахождения файла внутри экспортированной файловой системы. Проверка в поддеревьях иногда может приводить к возникновению проблем, если запрошенный файл был переименован в тот момент, когда он был открыт клиентом. Если вы считаете, что у вас могут возникнуть такие ситуации, используйте опцию no_subtree_check.

Опция secure_locks требует авторизации и аутентификации при блокировке файлов. Некоторые клиенты NFS не посылают мандаты вместе с запросами на блокировку и не воспринимают опцию secure_locks. В таких случаях у вас будет возможность заблокировать лишь те файлы, которые доступны для чтения всем без исключения. Наилучшим выходом из этой ситуации будет замена этих клиентов такими клиентами, которые предоставляют свои мандаты должным образом. А в качестве временной меры можно определить опцию insecure_locks.

После внесения изменений в файл exports не забудьте выполнить команду exportfs -a, чтобы изменения вступили в силу.

Демон nfsd: файловый сервер

После того как демон mountd убедился в правильности клиентского запроса на монтирование, клиенту разрешается запрашивать различные операции над файлами. Эти запросы обрабатываются на сервере демоном nfsd[2]. Он не должен выполняться на компьютере-клиенте NFS. Единственное исключение — случай, когда клиент сам экспортирует файловые системы.

Демон nfsd принимает числовой аргумент, определяющий, сколько экземпляров самого себя нужно породить посредством системного вызова fork. Правильный выбор числа процессов очень важен, но, к сожалению, относится к области черной магии. Если это число будет слишком мало или слишком велико, производительность сервера NFS может снизиться.

В общем случае восемь потоков nfsd достаточно для сервера, который используется нечасто, и достаточно для того, чтобы не возникало проблем с производительностью. На крупном сервере показатель должен быть в диапазоне от 12 до 20. Если вы заметите, что результаты выполнения команды ps чаще всего показывают nfsd в состоянии “D”, и что у вас всегда свободен некоторый объем ресурсов ЦП, увеличьте количество потоков. Увеличивайте число процессов до тех пор, пока показатель средней загруженности

[2] Демон nfsd всего лишь осуществляет не предусматривающий ответа системный вызов, код которого встроен в ядро.

сервера (его сообщает команда uptime) не начнет заметно возрастать. Это признак того, что система не справляется с работой. Уничтожьте несколько процессов, и все стабилизируется. Следует также регулярно запускать утилиту nfsstat, которая позволит выявлять проблемы, связанные с количеством потоков демона nfsd. Подробнее об этой утилите рассказывается в разделе 16.4.

На интенсивно эксплуатируемом сервере NFS с большим числом UDP-клиентов буферы UDP-сокетов могут переполниться, если запросы начнут поступать в то время, когда все демоны nfsd уже задействованы. Число переполнений можно узнать с помощью команды netstat -s. Запускайте дополнительные демоны до тех пор, пока количество переполнений не упадет до нуля, поскольку переполнение свидетельствует о серьезной нехватке серверных демонов.

Для изменения числа запускаемых демонов nfsd следует отредактировать соответствующий сценарий в каталоге /etc/init.d или определить это число в командной строке при запуске nfsd вручную (см. табл. 16.1).

16.3. Клиентская часть NFS

Процесс монтирования сетевых и локальных файловых систем во многом схож. Команда mount понимает запись вида *имя_компьютера*:*каталог* как путь к каталогу, расположенному на указанном компьютере. Этому каталогу будет поставлен в соответствие каталог локальной файловой системы. По завершении монтирования доступ к сетевой файловой системе осуществляется традиционными средствами. Таким образом, команда mount и ее NFS-расширения — самая важная для системного администратора на NFS-клиенте.

Для того чтобы файловую систему NFS можно было монтировать, ее необходимо соответствующим образом экспортировать (об этом рассказывалось выше). Клиентская команда showmount позволяет клиенту проверить, правильно ли сервер экспортирует файловые системы:

```
% showmount -e coyote
Export list for coyote:
/home/boggs inura.toadranch.com
```

Как следует из этого примера, каталог /home/boggs на сервере coyote экспортирован в клиентскую систему inura.toadranch.com. К команде showmount следует прибегать в первую очередь, если сетевая файловая система по какой-то причине не работает. Возможно, вы просто забыли выполнить команду exportfs -a после редактирования файла exports.

Если каталог был правильно экспортирован на сервере, а команда showmount сообщает об ошибке или возвращает пустой список, внимательно проверьте, все ли необходимые процессы запущены на сервере (portmap, mountd, nfsd, statd и lockd), разрешен ли в файлах hosts.allow и hosts.deny доступ к этим демонам и, вообще, та ли это клиентская система.

Реальное монтирование файловой системы осуществляется примерно такой командой:

```
# mount -o rw,hard,intr,bg coyote:/home/boggs /coyote/home/boggs
```

Флаги, указанные после опции -o, говорят о том, что файловая система монтируется в режиме чтения/записи, файловые операции разрешается прерывать, а повторные попытки монтирования должны делаться в фоновом режиме. Все эти параметры являются довольно типичными. Наиболее распространенные флаги приведены в табл. 16.4.

Таблица 16.4. Флаги монтирования NFS

Флаг	Назначение
`rw`	Монтирование файловой системы для чтения/записи (она должна экспортироваться сервером в режиме чтения/записи)
`ro`	Монтирование файловой системы только для чтения
`bg`	Если смонтировать файловую систему не удается (сервер не отвечает), следует перевести операцию в фоновый режим и продолжить обработку других запросов на монтирование
`hard`	Если сервер отключился, операции, которые пытаются получить к нему доступ, блокируются до тех пор, пока сервер не включится вновь
`soft`	Если сервер отключился, операции, которые пытаются получить к нему доступ, завершаются выдачей сообщения об ошибке. Этот флаг полезно устанавливать для того, чтобы предотвратить зависание процессов в случае неудачного монтирования не очень важных файловых систем
`intr`	Позволяет прерывать с клавиатуры заблокированные операции (будут выдаваться сообщения об ошибке)
`nointr`	Не позволяет прерывать с клавиатуры заблокированные операции
`retrans=`*n*	Указывает, сколько раз нужно повторить запрос, прежде чем будет выдано сообщение об ошибке (для файловых систем, смонтированных с флагом `soft`)
`timeo=`*n*	Задает интервал тайм-аута для запросов (в десятых долях секунды)
`rsize=`*n*	Задает размер буфера чтения равным *n* байтов
`wsize=`*n*	Задает размер буфера записи равным *n* байтов
`nfsvers=`*n*	Задает версию протокола NFS: 2 или 3 (обычно определяется автоматически)
`tcp`	Выбирает TCP в качестве транспортного протокола; по умолчанию принят протокол UDP.
`async`	Заставляет сервер отвечать на запросы на запись, прежде чем будет производиться запись данных на диск

Файловые системы, смонтированные с флагом `hard` (установка по умолчанию), могут вызывать зависание процессов при отключении сервера. Это особенно неприятно, когда такими процессами оказываются стандартные демоны. Вот почему не рекомендуется обслуживать ключевые системные программы через NFS. В общем случае использование флагов `soft` и `intr` позволяет сократить число проблем, связанных с NFS. Но эти же флаги могут вызывать нежелательные побочные эффекты (например, останов 20-часового процесса моделирования из-за временного сбоя в сети после 18 часов работы)[3]. Некоторыми средствами для борьбы с проблемами монтирования, такими как `autofs` (о нем речь пойдет далее в этой главе) располагают системы автоматического монтирования.

Размеры буферов чтения и записи применимы в отношении обоих протоколов — TCP и UDP, но оптимальные значения различны. В случае TCP буфер должен быть

[3] Джефф Форис (Jeff Forys), один из наших рецензентов, заметил по этому поводу: “Большинство сетевых файловых систем должно монтироваться с флагами `hard`, `intr` и `bg`, поскольку они наилучшим образом соответствуют исходной концепции NFS (надежность и отсутствие информации о состоянии). Флаг `soft` — это мерзкий сатанинский трюк! Если пользователь захочет прервать операцию монтирования, пусть он сделает это сам. В противном случае следует дождаться включения сервера, и все, в конце концов, нормализуется без какой бы то ни было потери данных”.

большим, поскольку данные передаются эффективнее. Хорошее значение — 32 Кбайт. В случае UDP, если сервер и клиент находятся в одной сети, оптимальный размер равен 8 Кбайт[4]. Тем не менее, по умолчанию принят размер 1 Кбайт, хотя даже на man-странице рекомендуется повысить это значение до 8 Кбайт.

Здесь следует сделать важное замечание. В устаревших ядрах Linux версий 2.2 и 2.4 стандартный размер входной очереди составляет 64 Кбайт. Когда на сервере NFS одновременно выполняется восемь экземпляров (потоков) демона nfsd, каждый из демонов сможет одновременно обрабатывать всего один запрос, а "лишние" запросы будут удаляться. Следовательно, имеет смысл увеличить размер входной очереди, но только для демона nfsd. По завершении его работы необходимо восстановить прежний размер, чтобы это изменение не отразилось негативно на других процессах. Изменить размер входной очереди можно в сценариях запуска системы, воспользовавшись средствами файловой системы /proc.

Так, в следующем примере размер очереди устанавливается равным 256 Кбайт, что является довольно удобным стандартным значением:

```
rmem_default='cat /proc/sys/net/core/rmem_default'
rmem_max='cat /proc/sys/net/core/rmem_max'
echo 262144 > /proc/sys/net/core/rmem_default
echo 262144 > /proc/sys/net/core/rmem_max
```

Далее запускается или перезапускается демон rpc.nfsd, после чего восстанавливаются первоначальные установки:

```
echo $rmem_default > /proc/sys/net/core/rmem_default
echo $rmem_max > /proc/sys/net/core/rmem_max
```

Протестировать точку монтирования можно с помощью команды df, как если бы это была обычная локальная файловая система:

```
% df /coyote/home/boggs
Filesystem           1k-blocks   Used  Available  Use% Mounted on
coyote:/home/boggs  17212156  1694128  14643692   11% /coyote/home/boggs
```

Разделы NFS размонтируются командой umount. Если сетевая файловая система кем-то используется в момент размонтирования, будет выдано сообщение об ошибке:

```
umount: /coyote/home/boggs: device is busy
```

Подобно другим файловым системам, NFS запрещает подобное размонтирование. Найдите с помощью команды lsof процессы, у которых есть открытые файлы в этой файловой системе, и уничтожьте эти процессы либо смените каталоги в случае интерпретаторов команд. Если ничего не помогает или сервер отключен, воспользуйтесь командой umount -f для принудительного размонтирования.

Монтирование файловых систем NFS на этапе начальной загрузки

С помощью команды mount можно создавать лишь временные сетевые точки монтирования. Файловые системы, которые являются частью постоянной конфигурации,

[4] Если используется утилита iptables, потребуется добавить правило, разрешающее прием фрагментированных пакетов, поскольку значение 8 Кбайт превышает максимальный размер передаваемого блока для Ethernet. Следует учесть, что поддержка фрагментированных пакетов делает систему более уязвимой к атакам типа "отказ в обслуживании".

должны быть перечислены в файле /etc/fstab, для того чтобы они автоматически монтировались на этапе начальной загрузки. С другой стороны, они могут обрабатываться утилитой автоматического монтирования, например демоном automount или amd (подробнее об этом речь пойдет, начиная с раздела 17.6).

📖 Об autofs будет рассказано в разделе 16.16.

Следующие элементы файла fstab предназначены для монтирования файловых систем /home и /usr/local компьютеров coyote и inura:

```
# filesystem        mountpoint    fstype   flags           dump  fsck
coyote:/home        /coyote/home  nfs      rw,bg,intr,hard    0     0
inura:/usr/local    /usr/local    nfs      ro,bg,intr,soft    0     0
```

При добавлении элементов в файл fstab обязательно создавайте каталоги точек монтирования с помощью команды mkdir. Можно сделать так, чтобы изменения вступили в силу немедленно (без перезагрузки). Для этого следует выполнить команду mount -a -t nfs, которая смонтирует все файловые системы типа nfs, перечисленные в файле fstab.

📖 Подробнее о файле fstab рассказывалось в разделе 7.4.

В поле flags файла /etc/fstab задаются параметры точек монтирования NFS. Это те же параметры, которые указываются в командной строке mount.

Ограничение экспортирования в незащищенные порты

Клиентам NFS разрешается использовать любой TCP- или UDP-порт для подключения к серверу NFS. Однако серверы Linux могут требовать, чтобы запросы поступали из привилегированного порта (номер которого меньше чем 1024), если файловая система экспортируется с опцией secure (включена по умолчанию). Впрочем, в мире персональных компьютеров и настольных Linux-систем применение привилегированных портов не приводит к реальному повышению безопасности системы.

Linux-клиенты NFS следуют традиционному (и по-прежнему рекомендуемому) подходу: по умолчанию выбирается привилегированный порт, что позволяет избежать ненужных конфликтов. Если требуется разрешить запросы на монтирование, поступающие из непривилегированных портов, экспортируйте файловые системы с опцией insecure.

16.4. Команда nfsstat: отображение статистики NFS

Команда nfsstat отображает различные статистические данные, накапливаемые в NFS. Команда nfsstat -s выдает статистику процессов сервера NFS, а команда nfsstat -c отображает информацию, касающуюся клиентских операций. Например:

```
% nfsstat -s
Server rpc stats:
calls          badcalls badauth   badclnt   xdrcall
24314112       311      9         302       0
Server nfs v2:
getattr        null     setattr   root      lookup       readlink
8470054 34%    58   0%  55199  0% 0     0%  1182897  4%  917   0%
read           wrcache  link      create    remove       rename
```

```
6602409 27%     0     0%  7452   0% 61544  0% 46712    0%  11537  0%
write           symlink   mkdir      rmdir      readdir       fsstat
7785789 32%     744   0%  3446   0% 2539   0% 13614    0%  69201  0%
```

Приведенные результаты получены на нормально функционирующем сервере NFS. Если более трех процентов вызовов терпят неудачу, то это говорит о наличии проблем на NFS-сервере или в сети. Выполните команду `netstat -s`, чтобы посмотреть общую статистику работы сети. Возможно, проблема связана с удалением пакетов, сборкой фрагментов или переполнением сетевой очереди. Подробнее о применении команды `netstat` для отладки сети рассказывается в разделе 19.4.

Регулярное выполнение команд `nfsstat` и `netstat` и анализ выдаваемой ими информации помогут администратору выявлять возникающие в NFS проблемы раньше, чем с ними столкнутся пользователи.

16.5. Специализированные файловые серверы NFS

Быстрый и надежный файловый сервер — один из важнейших элементов вычислительной среды. Конечно, дешевле всего организовать файловый сервер на рабочей станции Linux, воспользовавшись имеющимися жесткими дисками. Но это не оптимальное решение с точки зрения производительности и удобства администрирования.

Уже более десяти лет на рынке предлагаются специализированные файловые серверы NFS. У них есть целый ряд преимуществ по сравнению с “самодельными” системами:

- они оптимизированы на выполнение операций с файлами и, как правило, обеспечивают наилучшую производительность при работе с NFS;
- по мере увеличения требований к хранилищу файлов можно легко наращивать мощности сервера, даже если придется обслуживать терабайтовые массивы данных и сотни пользователей;
- они более надежны, чем Linux-системы, благодаря усеченному набору программ, применению дополнительных устройств, обеспечивающих избыточность, и зеркальному дублированию информации на жестких дисках;
- они обычно реализуют доступ к файлам для клиентов как Linux, так и Windows, а иногда даже содержат встроенные Web- и FTP-серверы;
- чаще всего они проще в администрировании, нежели файловые серверы Linux;
- они обладают улучшенными средствами резервного копирования и контроля состояния, чем обычные Linux-системы.

На рынке существующих систем лидируют устройства компании Network Appliance, Inc. (`www.netapp.com`). Диапазон предложений — от малых до очень крупных систем, и цены вполне приемлемы. Компания EMC занимает нишу устройств высшего класса. Они выпускают хорошие продукты, но их цены могут испугать кого угодно. Приготовьтесь также изучить модные маркетинговые термины[5].

В наши дни популярность приобрели также серверы SAN (Storage Area Network – сеть хранения данных). Они отличаются от специализированных файловых серверов NFS тем, что они не умеют распознавать файловые системы; они просто обслуживают

[5] Пару слов о маркетинговых терминах. В рассматриваемом контексте часто можно встретить выражение “сетевое устройство хранения данных”, сокращенно NAS (Network Attached Storage). На человеческом языке это означает “файловый сервер”.

дисковые блоки. Судя по отзывам, они пока что малопригодны для использования. Их сложно интегрировать в существующую среду, и они требуют много времени на обслуживание.

16.6. Автоматическое монтирование

Индивидуальное монтирование файловых систем посредством упоминания их в файле `/etc/fstab` сопряжено в крупных сетях с целым рядом проблем.

Во-первых, ведение файла `/etc/fstab` на нескольких сотнях компьютеров — весьма утомительная задача. Каждый из компьютеров может отличаться от других и требовать особого подхода.

Во-вторых, если файловые системы монтируются с множества компьютеров, то в случае сбоя всего лишь одного из серверов наступает хаос, так как все команды, пытающиеся получить доступ к точкам монтирования этого сервера, зависают.

В-третьих, сбой какого-нибудь важного сервера может нанести немалый ущерб пользователям, сделав недоступными такие важные разделы, как, например, `/usr/share/man`. Самый простой выход из этой ситуации — временно смонтировать копию раздела с резервного сервера.

Демон автоматического монтирования подключает файловые системы, когда к ним делается обращение, и отключает их, когда надобность в них отпадает. Этот процесс осуществляется незаметно для пользователей и позволяет свести к минимуму количество активных точек монтирования. Демону можно также предоставить список "реплицированных" (идентичных) файловых систем, чтобы сеть могла функционировать в случае отказа основного сервера.

Для реализации фонового подключения и отключения демон связывает драйвер виртуальной файловой системы с каталогами, обозначенными как точки автоматического монтирования. Ранее он делал это, выступая в качестве сервера NFS, но такой подход имел ряд серьезных ограничений, поэтому редко применяется в современных системах. В настоящее время используется драйвер файловой системы `autofs`, встроенный в ядро.

Вместо того чтобы зеркально дублировать в сети реальную файловую систему, демон автоматического монтирования воссоздает ее иерархию в соответствии со спецификациями, указанными в файле конфигурации. Когда пользователь ссылается на каталог виртуальной файловой системы демона, последний перехватывает эту ссылку, монтирует реальную файловую систему, к которой обращается пользователь, и передает ссылку дальше.

`automount`: монтирование файловых систем по требованию

Идея автоматического монтирования была предложена компанией Sun. Имеющийся в Linux демон `autofs` имитирует работу одноименной утилиты компании Sun, хотя реализован независимо и обладает рядом отличительных особенностей.

Демон `automount` — это фоновый процесс, конфигурирующий заданную точку монтирования для драйвера файловой системы `autofs`, т.е. той части подсистемы автоматического монтирования, которая находится в ядре. Сценарий `/etc/init.d/autofs` анализирует главный файл (обычно `/etc/auto.master`) и запускает демон `automount` для каждой из перечисленных там точек монтирования.

Необходимость в ручном запуске демона возникает редко, так как почти все административные операции реализуются посредством сценария /etc/init.d/autofs (или, в случае Red Hat и Fedora, /etc/rc.d/init.d/autofs)[6]. Подобно большинству других сценариев запуска системы, он принимает один аргумент командной строки, в данном случае start, stop, reload, restart или status. Когда в конфигурацию подсистемы автоматического монтирования вносятся изменения, необходимо выполнять команду autofs reload, чтобы изменения вступили в силу. Команда autofs status позволяет получить информацию о существующих точках автоматического монтирования.

Файл auto.master связывает точку монтирования с так называемой *таблицей назначений* (map). В этой таблице имя каталога, к которому осуществляется доступ (называется *ключом*), транслируется в командную строку, благодаря которой команда mount осуществит реальное монтирование. Таблица назначений может храниться в текстовом или исполняемом файле, базе данных NIS или LDAP.

Когда пользователь ссылается на каталог, смонтированный через файловую систему autofs, демон automount получает сигнал о попытке доступа. Демон проверяет таблицу назначений, чтобы выяснить, какую файловую систему следует монтировать, после чего осуществляет монтирование и возвращает управление пользователю.

Получить список автоматически монтируемых файловых систем и связанных с ними демонов automount можно с помощью команд mount и ps.

```
% mount
/dev/hda3 on / type ext2 (rw)
proc on /proc type proc (rw)
/dev/hda1 on /boot type ext2 (rw)
automount(pid8359) on /misc type autofs     // файловая система
                                            // демона автоматического монтирования
     (rw,fd=5,pgrp=8359,minproto=2,maxproto=4)
automount(pid8372) on /net type autofs      // файловая система
                                            // демона автоматического монтирования
     (rw,fd=5,pgrp=8372,minproto=2,maxproto=4)

% ps auxw | grep automount
root    8359  0.0  1.0  1360  652 ?     S    Dec27  0:00
     /usr/sbin/automount /misc file /etc/auto.misc
root    8372  0.0  1.0  1360  652 ?     S    Dec27  0:00
     /usr/sbin/automount /net program /etc/auto.net
```

Здесь показаны две файловые системы типа autofs, смонтированные в каталогах /misc и /net. Эти виртуальные файловые системы связаны с процессами automount, идентификаторы которых 8359 и 8372, соответственно. Строки запуска демона automount, указанные в сценарии /etc/init.d/autofs, отображаются в выводе команды ps. Файл auto.misc, в котором хранится таблица назначений, — это обычный текстовый файл, тогда как файл auto.net является исполняемым. Формат этих файлов будет описан ниже.

Главный файл

Файл /etc/auto.master содержит список каталогов, к которым должны монтироваться файловые системы типа autofs, и соответствующих им таблиц назначений. Для

[6] Не путайте сценарий autofs с одноименной файловой системой. Их объединяет то, что сценарий сообщает ядру, как конфигурировать систему.

каждой таблицы указывается не только имя и каталог, где она находится, но также дополнительные флаги, понимаемые командой `mount` (относятся к опции `-o` ее командной строки). Эти флаги применяются ко всем записям таблицы. Кроме того, следует подчеркнуть, что флаги, указанные в главном файле, объединяются с опциями таблицы назначений; оба набора параметров передаются команде `mount`. Это особенность Linux.

Простейший главный файл, использующий показанную в следующем подразделе таблицу назначений, будет иметь примерно такой вид:

```
# Directory     Map               Options
/chimchim       /etc/auto.chim    -secure,hard,bg,int
```

Главный файл может быть замещен или дополнен своей версией, совместно используемой через систему NIS. Источник информации для подсистемы автоматического монтирования указывается в поле `automount` файла `/etc/nsswitch.conf`. Подробнее об этом файле рассказывается в разделе 17.4.

Таблицы назначений

Таблицы назначений (называемые в некоторых системах таблицами косвенных назначений — “indirect maps”) используются для автоматического монтирования нескольких файловых систем в общем каталоге. Имя каталога задается в главном файле, а не в самой таблице. Например, таблица назначений для файловых систем, монтируемых в каталоге `/chimchim` (в соответствии с показанным выше главным файлом), может иметь следующий вид:

```
users       chimchim:/chimchim/users
devel       -soft,nfsvers=3 chimchim:/chimchim/devel
info        -ro chimchim:/chimchim/info
```

В первой колонке указывается имя подкаталога, в котором будет смонтирована файловая система. В следующих колонках перечислены опции монтирования и приведен исходный путь к файловой системы. В данном примере (файл `/etc/auto.chim`) демону `automount` сообщается о том, что он может монтировать каталоги `/chimchim/users`, `/chimchim/devel` и `/chimchim/info` с компьютера `chimchim`, при этом каталог `info` монтируется только для чтения, а каталог `devel` монтируется в режиме `soft` с использованием протокола NFS версии 3.

В рассматриваемой конфигурации подкаталоги на компьютере `chimchim` и на локальном компьютере будут идентичными, хотя это вовсе не обязательно.

Исполняемые файлы таблиц

Если файл, содержащий таблицу косвенных назначений, является исполняемым, то он считается сценарием (или программой), динамически генерирующим информацию об автоматическом монтировании. Демон не читает таблицу в текстовом виде, а запускает файл на выполнение, передавая ему аргумент (“ключ”), где указывается, к какому подкаталогу пользователь пытается получить доступ. Сценарий отвечает за вывод соответствующей записи таблицы. Если переданный ключ неверен, сценарий завершается, ничего не отображая.

Такая методика очень удобна и позволяет компенсировать многочисленные недостатки довольно странной системы конфигурирования демона `automount`. По сути, она дает возможность создать единый для всей организации конфигурационный файл произвольного формата. Можно написать несложный сценарий на Perl, декодирующий глобальные

конфигурационные параметры на каждом компьютере. В состав некоторых систем входит удобный сценарий `/etc/auto.net`, принимающий в качестве ключа имя компьютера и монтирующий все экспортируемые файловые системы этого компьютера.

Демон `automount` обладает одной сбивающей с толку характеристикой, о которой необходимо упомянуть. Когда вы перечисляете содержимое родительского каталога в автоматически смонтированной файловой системе, то этот каталог будет отображаться как пустой, независимо от того, сколько в нем было смонтировано файловых систем. Автоматически смонтированные файловые системы нельзя просмотреть с помощью обозревателя с графическим интерфейсом пользователя. В доказательство можно привести такой пример:

```
$ ls /portal
$ ls /portal/photos
art_class_2004      florissant_1003           rmnp03
blizzard2003        frozen_dead_guy_Oct2004   rmnp_030806
boston021130        greenville.021129         steamboat2002
```

Файловая система `photos` на самом деле "жива и здорова" и смонтирована в каталоге `/portal`. Доступ к ней возможен через указание полного путевого имени. Тем не менее, обзор каталога `/portal` не дает никакой информации о ее существовании. Если бы эта файловая система была смонтирована посредством файла `/etc/fstab` или вручную с помощью команды `mount`, то она бы вела себя подобно любому другому каталогу и была бы видима как член родительского каталога.

Чтобы избежать этой проблемы, можно, например, создать теневой каталог, содержащий символические ссылки на точки автоматического монтирования. Например, если `/automounts/photos` является символической ссылкой на каталог `/portal/photos`, то с помощью команды `ls /automounts` можно узнать, что `photos` – это автоматически смонтированный каталог. Ссылками на `/automounts/photos` по-прежнему управляет демон `automount`, и все они работают нормально. К сожалению, эти символические ссылки нужно постоянно обслуживать, а если их периодически не переопределять с помощью сценария, то они могут оказаться ошибочными.

16.7. Рекомендуемая литература

- Callaghan, Brent. *NFS Illustrated*. Addison-Wesley. 1999.
- Stern, Hal, Mike Eisler, and Ricardo Labiaga. *Managing NFS and NIS, Second Edition*. Sebastopol: O'Reilly & Associates. 2001.

В табл. 16.5 перечислены документы RFC, посвященные протоколу NFS и его расширениям.

Таблица 16.5. Документы RFC, связанные с NFS

RFC	Название	Дата
1094	Network File System Protocol Specification	Май 1989 г.
1813	NFS Version 3 Protocol Specification	Июнь 1995 г.
2623	NFS Version 2 and Version 3 Security Issues	Июнь 1999 г.
2624	NFS Version 4 Design Considerations	Июнь 1999 г.
3530	NFS Version 4 Protocol	Апрель 2003 г.

16.8. Упражнения

☆ 16.1. Проанализируйте локальную конфигурацию NFS в своей системе. Используется ли NFS или применяется какое-то другое решение? Используется ли автоматическое монтирование? На какие компромиссы пришлось пойти, создавая такую конфигурацию?

☆ 16.2. Какова взаимосвязь между демонами `mountd`, `nfsd` и `portmap`? Как зависимость от демона `portmap` сказывается на безопасности системы?

☆☆ 16.3. Каковы побочные эффекты того, что в NFS не поддерживается понятие сеанса? Как это влияет на блокировки файлов, права доступа и безопасность? Если бы сетевая файловая система хранила информацию о состоянии, то чем она отличалась бы от NFS?

☆☆ 16.4. Ваш начальник желает экспортировать вам свои каталоги `/usr` и `/usr/local` через NFS. Ответьте на следующие вопросы.

а) Предположим, вследствие существующих ограничений вы хотите, чтобы служащие только вашего отдела (подсеть 192.168.123.0/24) могли работать с этими файловыми системами. Какие строки необходимо добавить и в какие файлы, чтобы такая конфигурация стала возможной? Обратите внимание на выбор опций экспортирования.

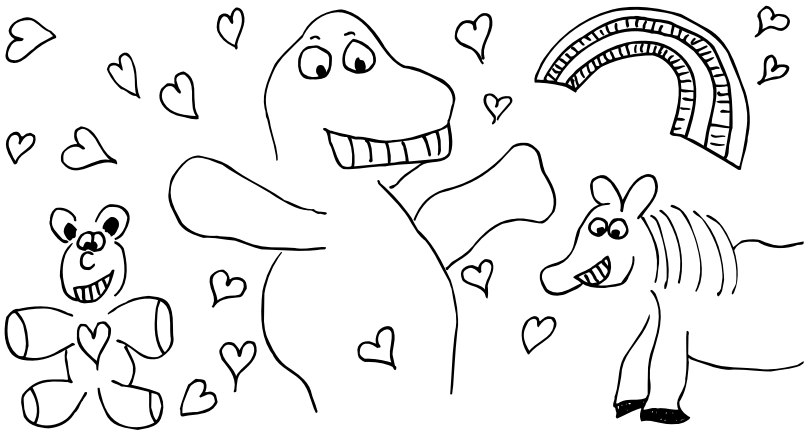
б) Какие действия следует предпринять для того, чтобы демоны `mountd` и `nfsd` смогли распознать эти новые файловые системы? Как проверить факт совместного использования каталогов, не монтируя их?

в) Какой стратегии следует придерживаться, чтобы все компьютеры локальной подсети автоматически монтировали экспортируемые каталоги с использованием точек монтирования `/mnt/usr` и `/mnt/usr/local`?

Book_LinuxAdmHand.indb 562 17.04.2007 13:55:42

Глава 17

Совместное использование системных файлов



Нормально функционирующая система зависит от десятков, а иногда даже сотен файлов конфигурации, каждый из которых должен содержать корректные данные. Умножьте количество конфигурационных файлов, имеющихся на одном компьютере, на число компьютеров в сети, и вы получите тысячи файлов — слишком много, чтобы управлять ими вручную.

Однако на практике многие компьютеры очень часто схожи друг с другом с точки зрения администрирования. Вместо того чтобы редактировать текстовые файлы на каждом компьютере, гораздо эффективнее будет объединить компьютеры в группы, совместно использующие конфигурационные данные. Это можно сделать несколькими способами.

Простейший из них заключается в хранении главной копии каждого конфигурационного файла в каком-то одном месте и распространении этой копии членам группы при каждом изменении. Преимуществом такого решения является простота и доступность во всех Linux- и UNIX-системах.

Другой подход состоит в том, чтобы удалить все текстовые файлы и заставить каждый компьютер получать информацию о своей конфигурации с центрального сервера. Делать это сложнее, чем просто копировать файлы, но зато попутно решаются другие проблемы. Например, клиенты всегда будут получать исправленные версии файлов, даже если во время внесения изменений клиентский компьютер был отключен. Кроме того, загружать информацию с сервера в некоторых случаях можно быстрее, чем из фай-

ла (это зависит от быстродействия локального диска и объема кэширования, выполняемого сервером). Правда, при такой схеме отказ сервера может привести к зависанию всей сети.

Сегодня задача усложняется еще и тем, что во многих организациях устанавливаются компьютеры, работающие на разных платформах: часть из них работает под управлением UNIX, другая часть — под управлением Linux, а некоторые компьютеры и вовсе работают в среде Windows. Разумеется, пользователи, которым приходится привыкать к тому, что, например, на каждом из таких компьютеров нужно вводить разные пароли, испытывают из-за этого неудобства. Синхронизация конфигурации и сведений о пользователях в разных системах (например, в Linux и Windows) поначалу была несбыточной мечтой. Сегодня это уже обычное явление.

Попытки разработать распределенные административные базы данных для больших сетей продолжались несколько десятилетий, в результате чего были созданы интересные системы. К сожалению, ни одна из этих систем сегодня не кажется нам безупречной по идеологии. Некоторые из них просты, но не отличаются достаточной степенью безопасности и масштабирования. Другие обладают хорошими функциональными возможностями, но чересчур тяжеловесны. Всем этим системам присущи ограничения, которые могут помешать организовать и настроить сеть так, как нужно администратору, и ни одна из систем не позволяет управлять всей информацией, которую требуется совместно использовать на разных компьютерах.

В этой главе сначала будет рассмотрен ряд базовых методик синхронизации файлов в сети. Затем мы поговорим о наиболее широко используемой административной СУБД — NIS, которая первоначально была разработана для работы в среде UNIX. В конце главы речь пойдет об LDAP — более сложной системе, не зависящей от платформы, которая уже де-факто стала стандартом. Сегодня во многих организациях постепенно переходят на использование LDAP. К этому их подталкивает, прежде всего, принятие фирмой Microsoft стандарта LDAP в ее продукте Active Directory, а также желание добиться более качественной интеграции сред Linux и Windows.

17.1. Предмет совместного использования

В Linux существует много файлов конфигурации, но далеко не все из них имеет смысл совместно использовать на нескольких компьютерах. Наиболее распространенные случаи перечислены в табл. 17.1.

Таблица 17.1. Системные файлы, которые часто являются объектами совместного использования

Имя файла	Назначение/содержимое
`/etc/passwd`	База данных с информацией о пользовательских учетных записях
`/etc/shadow`[а]	Файл скрытых паролей
`/etc/group`	Определения Linux-групп
`/etc/hosts`	Соответствия между именами компьютеров и их IP-адресами
`/etc/networks`[б]	Соответствия между именами сетей и их IP-адресами
`/etc/services`	Перечень номеров портов для известных сетевых служб
`/etc/protocols`	Соответствия между символьными именами и номерами протоколов
`/etc/ethers`	Соответствия между именами компьютеров и их Ethernet-адресами

Окончание табл. 17.1

Имя файла	Назначение/содержимое
`/etc/mail/aliases`	Псевдонимы электронной почты
`/etc/rpc`	Перечень идентификаторов RPC-служб
`/etc/netgroup`	Определения групп компьютеров, пользователей и сетей
`/etc/printcap`	База данных с информацией о принтере
`/etc/printcap.cups`	База данных с информацией о принтере (альтернативный путь)
`/etc/termcap`	Информационная база данных о типе терминала

[а] Совместное использование может не поддерживаться другими версиями UNIX из-за несовпадения механизмов шифрования (см. раздел 6.1).

[б] Используется не во всех системах.

Другие конфигурационные файлы связаны с конкретными приложениями (например, файл `/etc/sendmail.cf` для `sendmail`) и не поддерживаются административными СУБД, такими как NIS и LDAP. Такие файлы приходится копировать.

Доступ к файлам, перечисленным в табл. 17.1, традиционно осуществляется через функции стандартной библиотеки языка C. Например, поиск в файле `/etc/passwd` выполняют функции `getpwuid`, `getpwnam` и `getpwent`. Они берут на себя открытие, чтение и синтаксический анализ файла `passwd`, освобождая от этой задачи программы пользовательского уровня. В современных дистрибутивах Linux используются также модули PAM (Pluggable Authentication Module — подключаемый модуль аутентификации), которые реализуют стандартный программный интерфейс безопасного поиска информации. Эти модули позволяют легко интегрировать в Linux различные системы, в частности Kerberos и LDAP.

📖 О модулях PAM речь пойдет в конце раздела 20.5.

Наличие административных СУБД усложняет поиск информации, поскольку появляется несколько альтернативных источников. Традиционные библиотечные функции (`getpwent` и др.) "знают" о существовании основных СУБД и умеют обращаться к ним в дополнение к обычным текстовым файлам (или вместо них). Точный порядок поиска устанавливается системным администратором (подробнее об этом — в разделе 17.4).

17.2. Демон `nscd`: кэширование результатов поиска

В некоторых дистрибутивах большую роль в обработке системных файлов играет демон `nscd`, название которого немного сбивает с толку (name service cache daemon — демон кэширования для службы имен). Этот демон работает в связке с библиотекой языка C, кэшируя результаты таких функций, как `getpwent` и ей подобных. По сути, демон является просто оболочкой этих функций; ему не известно, какие источники данных опрашивались. В теории демон должен способствовать повышению эффективности поиска информации в системных файлах, но с точки зрения пользователей реальный эффект практически не ощутим.

Мы говорим, что название "демон кэширования для службы имен" способно ввести в заблуждение, поскольку термин "служба имен" обычно относится к DNS, распределенной СУБД, осуществляющей связь между доменными именами и IP-адресами.

Демон nscd в числе прочего кэширует и результаты DNS-запросов (одна из контролируемых им функций — gethostbyname), но он также связан с функциями, черпающими информацию из файлов passwd и group и их сетевыми эквивалентами (по соображениям безопасности результаты запросов к файлу /etc/shadow не кэшируются).

О DNS рассказывалось в главе 15.

В принципе, демон nscd не должен влиять на работу системы; он лишь ускоряет выполнение повторяющихся операций поиска. Но на практике влияние демона может оказаться непредсказуемым из-за того, что он хранит собственную копию результатов поиска. Эти данные находятся в кэше в течение фиксированного интервала времени (задается в конфигурационном файле демона — /etc/nscd.conf), поэтому всегда есть вероятность того, что самые последние изменения не будут отражены в кэше, пока прежние данные не устареют. Вообще-то демон достаточно "умен" и следит за состоянием локальных источников данных (например, файлом /etc/passwd), поэтому локальные изменения должны вступать в силу в течение 15 секунд. Что касается удаленных источников, например NIS, то в худшем случае величина задержки будет равна периоду устаревания.

Мы с вами упоминали несколько примеров дистрибутивов. Так вот, только SUSE запускает nscd по умолчанию. В Fedora и RHEL демон nscd инсталлируется, но не запускается на этапе загрузки системы по умолчанию; чтобы разрешить использование демона nscd, нужно выполнить команду chkconfig nscd on. Debian и Ubuntu разрешают работу демона nscd, но не инсталлируют его по умолчанию; чтобы его загрузить, нужно выполнить команду apt-get install nscd.

Демон nscd запускается на этапе начальной загрузки и работает непрерывно. По умолчанию в файле /etc/nscd.conf задан период 10 минут для файла passwd и 1 час для файлов hosts и group. Интервал между попытками в случае неудачного запроса составляет 20 секунд. Эти значения редко приходится менять. Если вас удивляет, что сделанные вами изменения не вступили в силу, то, очевидно, виной тому демон nscd.

17.3. Копирование файлов

Простое копирование файлов не является элегантным решением, но оно применимо на всех типах компьютеров и отличается простотой настройки и сопровождения. Оно также надежно, поскольку число перекрестных зависимостей между компьютерами сводится к минимуму (правда, при этом легче нарушить синхронизацию систем). Благодаря этому мы всегда вольны решать, что распространять по сети и как именно. У нас появляется возможность своевременно обновлять не только системные файлы, но и приложения вместе с файлами данных.

Довольно большое число конфигурационных файлов не поддерживается ни одной из административных СУБД (среди них файл /etc/ntp.conf, задающий правила сетевой синхронизации часов, и файл /etc/sendmail.cf, определяющий порядок доставки почты программой sendmail). Когда необходимо поддерживать согласованность этих файлов, не остается другого выхода, кроме как копировать их, даже если другая информация распространяется посредством NIS или LDAP.

Системы копирования файлов могут работать по модели принудительной рассылки либо модели рассылки по запросу. В первом случае главный сервер периодически отправляет самые свежие версии файлов каждому клиенту независимо от желания последнего. Файлы могут копироваться явно при каждом изменении или просто регулярно вы-

сылаться согласно графику (при этом, вероятно, одни файлы будут обновляться чаще других).

Преимущество модели принудительной рассылки заключается в том, что система распространения работает централизованно, на одном компьютере. Файлы, списки клиентов, сценарии обновления и расписания хранятся в одном месте, что делает эту схему простой в управлении. Есть у нее и недостаток: каждый клиент должен позволять главному серверу модифицировать свои системные файлы, что не всегда приемлемо с точки зрения безопасности.

В модели рассылки по запросу каждый клиент отвечает за обновление самого себя с помощью данных, запрашиваемых с сервера. Это менее централизованный способ распространения файлов, зато более гибкий и безопасный. Данная схема копирования особенно привлекательна, когда совместно используемые данные находятся в разных административных областях, поскольку вовсе не обязательно, чтобы главный сервер и компьютер-клиент располагались в одной области.

Утилита `rdist`: принудительная рассылка файлов

Проще всего распространять файлы с центрального сервера с помощью утилиты `rdist`. Она немного напоминает программу `make`: вначале пользователь в текстовом редакторе составляет описание (спецификацию) файлов, подлежащих рассылке, а затем утилита `rdist` осуществляет операции копирования в соответствии с описанием. Утилита копирует файлы только в том случае, если их копии устарели, поэтому в спецификации можно задать копирование всех файлов, а утилита `rdist` самостоятельно определит, какие файлы и когда следует копировать.

Утилита `rdist` сохраняет информацию о владельце, группе, правах доступа и времени модификации файла. Когда утилита обновляет существующий файл, то перед инсталляцией новой версии она удаляет старую. Это позволяет использовать данную утилиту для пересылки исполняемых файлов, запущенных во время обновления[1].

Традиционно утилита `rdist` работала поверх команды `rsh` и пользовалась ее средствами аутентификации для получения доступа к удаленным системам. К сожалению, она имела ряд слабых мест с точки зрения безопасности и по умолчанию была отключена во многих современных операционных системах. Несмотря даже на то, что в документации к `rdist` до сих пор упоминается `rsh`, не нужно думать, что `rsh` является приемлемым вариантом.

Современные версии утилиты `rdist` примечательны тем, что позволяют заменять команду `rsh` любой другой, понимающей аналогичный синтаксис. Чаще всего это команда `ssh`, которая использует шифрование с открытым ключом для аутентификации хостов и, кроме того, шифрует весь диалог, не давая возможности злоумышленникам, прослушивающим сеть, получать копии системных файлов. Недостаток заключается в том, что удаленные серверы `ssh` должны работать в режиме, позволяющем не указывать пароль (хотя клиента нужно аутентифицировать с помощью пары криптографических ключей), а это менее безопасный режим, чем требуется в обычных условиях. Но все равно это значительный шаг вперед по сравнению с `rsh`. Подробнее о демоне `sshd` и его режимах аутентификации рассказывается в разделе 20.11.

[1] Хотя старая версия удаляется из пространства имен файловой системы, она продолжает существовать до тех пор, пока не освобождены все ссылки на нее. Об этом эффекте следует помнить при работе с журнальными файлами. Подробнее об этом рассказывалось в конце раздела 10.1.

Итак, разобравшись с тем, какую угрозу таит в себе утилита rdist, рассмотрим в деталях, как она функционирует. Подобно программе make, утилита rdist ищет в текущем каталоге управляющий файл (Distfile или distfile). Команда rdist -f *файл* задает имя управляющего файла явно. В качестве разделителей в этом файле используются знаки табуляции, пробелы и символы новой строки. Комментарии предваряются знаком #.

Тело управляющего файла состоит из инструкций следующего вида:

```
метка: имена_файлов -> адресаты команды
```

Поле *метка* является именем инструкции. В командной строке можно ввести команду rdist *метка*, которая обеспечит рассылку только файлов, упомянутых в указанной инструкции.

Поля *имена_файлов* и *адресаты* — это, соответственно, список файлов, подлежащих копированию, и список компьютеров, куда их нужно переслать. Если в списке больше одного элемента, его нужно заключить в круглые скобки, а элементы — разделить пробелами. Список имен файлов может включать метасимволы, допустимые в интерпретаторе команд (например, /usr/man/man[123] или /usr/lib/*). Допустима также запись ~*пользователь*, но соответствующие ей значения на хосте-отправителе и хосте-адресате могут не совпадать.

По умолчанию утилита rdist копирует файлы и каталоги, перечисленные в списке имен, в эквивалентные каталоги на каждом хосте-адресате. Этот порядок можно изменить, указав последовательность команд. Каждую команду следует завершать точкой с запятой.

Поддерживаются такие команды:

```
install опции [целевой_каталог];
notify список_адресов;
except список_имен;
except_pat список_шаблонов;
special [список_имен] строка;
cmdspecial [список_имен] строка;
```

Команда install задает опции, влияющие на то, как утилита rdist копирует файлы. Опции, как правило, используются для обработки символических ссылок, проверки правильности применяемого алгоритма сравнения файлов, а также для контроля над процедурой обработки файлов, отсутствующих в исходном дереве каталогов. Опции, перед которыми должен стоять флаг -o, включают список разделенных запятыми имен. Например, строка

```
install -oremove,follow ;
```

заставляет утилиту rdist отслеживать символические ссылки (а не копировать их как обычные файлы) и удалять на целевом компьютере файлы, для которых не найдено соответствие на исходном компьютере. Полный список опций приведен на man-странице утилиты rdist. В большинстве случаев установки по умолчанию вполне подходят.

Название команды install слегка вводит в заблуждение, поскольку файлы копируются независимо от того, присутствует эта команда или нет. Опции задаются так, как они задавались бы в командной строке утилиты rdist, но при включении в управляющий файл они действуют только на совокупность файлов, указанную в инструкции.

Необязательный аргумент *целевой_каталог* задает каталог, в который инсталлируются файлы на компьютерах-адресатах. По умолчанию утилита rdist использует исходные имена каталогов.

В качестве аргумента команды notify задается список адресов электронной почты. При каждом обновлении очередного файла утилита rdist посылает по этим адресам почтовое сообщение. Ко всем адресам, не содержащим знак @, добавляется имя хоста-адресата. Например, при выдаче списка файлов, откорректированных на компьютере anchor, запись "pete" превратится в "pete@anchor".

Команды except и except_pat предназначены для удаления имен из списка файлов, подлежащих копированию. Аргументы команды except трактуются буквально, аргументы команды except_pat интерпретируются как регулярные выражения. Эти команды весьма полезны, поскольку утилита rdist, как и make, позволяет задавать макроконстанты в начале управляющего файла. Благодаря этому появляется возможность использовать один список файлов в нескольких инструкциях, указывая для каждого компьютера только добавления или удаления.

Команда special позволяет выполнить команду интерпретатора sh (аргумент *строка*, который следует брать в кавычки) на всех удаленных компьютерах. Если будет задан *список_имен*, rdist выполнит команду после копирования каждого из указанных файлов. Без этого списка rdist выполнит команду после каждого файла. cmdspecial работает подобным образом, но только выполняет команду интерпретатора sh после завершения каждой операции копирования. (Содержимое *списка_имен* передается интерпретатору sh в виде переменной окружения.)

Приведем пример файла Distfile:

```
SYS_FILES = (/etc/passwd /etc/group /etc/mail/aliases)
GET_ALL   = (chimchim lollipop barkadon)
GET_SOME  = (whammo spiff)

all: ${SYS_FILES} -> ${GET_ALL}
   notify barb;
   special /etc/mail/aliases "/usr/bin/newaliases";

some: ${SYS_FILES} -> ${GET_SOME}
   except /etc/mail/aliases;
   notify eddie@spiff;
```

В данной конфигурации три указанных системных файла копируются на компьютеры chimchim, lollipop и barkadon, а по адресу barb@*адресат* посылается сообщение с описанием всех изменений и замеченных ошибок. После копирования файла /etc/mail/aliases утилита rdist выполняет на каждом компьютере команду newaliases. На компьютеры whammo и spiff копируются только два файла, а отчет посылается по адресу eddie@spiff. Команда newaliases на этих двух компьютерах не выполняется.

📖 Подробнее о команде newaliases рассказывается в конце раздела 18.4.

Чтобы утилита rdist могла работать, нужно, чтобы демон sshd на принимающих хостах доверял хосту, осуществляющему рассылку файлов. Для этого на сервере нужно сгенерировать текстовый ключ и сохранить копию его открытой части в файле ~root/.ssh/authorized_keys на каждом клиенте. Имеет также смысл ограничить сферу применения ключа и количество хостов, от которых он может приниматься. Более подробно об этом речь пойдет в разделе 20.11 при описании метода Б.

Утилита rsync: более безопасная рассылка файлов

Утилита rsync, написанная Эндрю Триджеллом (Andrew Tridgell) и Полом Макеррасом (Paul Mackerras), подобна утилите rdist, но иначе реализована. Она не использует управляющий файл (хотя у сервера все же есть свой конфигурационный файл) и напоминает улучшенную версию команды rcp, пытающуюся сохранить ссылки, время модификации и права доступа. Утилита rsync более эффективна, чем rdist, поскольку анализирует отдельные файлы и пытается передавать только изменения между версиями. Эта утилита имеется в большинстве дистрибутивов Linux, хотя по умолчанию она может быть не инсталлирована.

📖 Утилита rsync доступна по адресу rsync.samba.org.

С нашей точки зрения, основным преимуществом утилиты rsync является то, что на принимающих компьютерах серверный процесс может запускаться из демона xinetd или inetd. Сервер (на самом деле это та же утилита rsync, но работающая в другом режиме; она должна быть инсталлирована как на главном компьютере, так и на клиентах) допускает гибкую настройку: он может предоставлять удаленный доступ лишь к заданному набору каталогов и требовать, чтобы главный компьютер подтвердил свою подлинность паролем. Поскольку доступ на уровне команды ssh не требуется, можно организовать распространение системных файлов посредством утилиты rsync, не жертвуя безопасностью. (Тем не менее, утилита rsync разрешает пользоваться командой ssh, а не серверным процессом, работающим под управлением демона inetd.) Более того, утилита способна работать в режиме рассылки по запросу (самостоятельно запрашивая файлы у сервера rsync и не дожидаясь принудительной рассылки), что делает ее еще более безопасной.

К сожалению, утилита rsync в целом не столь гибкая в настройке, как rdist, и ее конфигурационный файл не такой сложный, как файл distfile. На стороне клиента нельзя выполнить произвольную команду и нельзя рассылать файлы нескольким хостам одновременно.

Например, команда

```
# rsync -gopt --password-file=/etc/rsync.pwd /etc/passwd
      lollipop::sysfiles
```

посылает файл /etc/passwd на компьютер lollipop. Флаг -gopt указывает на сохранение прав доступа, идентификаторов владельцев и времени модификации файла. Два двоеточия в выражении lollipop::sysfiles заставляют утилиту связаться с удаленным сервером rsync непосредственно через порт 873, а не использовать для этого команду ssh. Для аутентификации соединения берется пароль из файла /etc/rsync.pwd[2].

В этом примере передается только один файл, хотя утилита rsync способна обрабатывать одновременно целую группу файлов. Кроме того, флаги --include и --exclude позволяют указать список регулярных выражений, с которым будут сравниваться имена файлов. Благодаря этому можно сформировать довольно сложный набор правил копи-

[2] Пароль не посылается по сети в текстовом виде, но сами передаваемые файлы не шифруются. С другой стороны, если в качестве транспортного средства используется команда ssh (rsync -gopt -e ssh /etc/passwd /etc/shadow lollipop:/etc — обратите внимание на одинарное двоеточие), то шифруется все соединение, однако демон sshd придется сконфигурировать так, чтобы он не запрашивал пароль. Воистину, мы сами выбираем свою казнь!

рования. Если командная строка становится слишком громоздкой, поместите регулярные выражения в отдельные файлы. Для этого существуют опции `--include-file` и `--exclude-file`.

После того как пакет `rsync` будет инсталлирован, Red Hat, Fedora и SUSE обеспечат конфигурацию `xinetd` этой утилиты. А чтобы активировать работу сервера, вам нужно будет отредактировать файл `/etc/xinetd.rsync`, поменяв строку `disable=yes` на `disable=no`.

Чтобы активизировать сервер в Debian и Ubuntu, вам придется самостоятельно добавить соответствующую запись в файл `/etc/inetd.conf`. Если предположить, что вы используете пакет TCP-оболочек, то такая запись может выглядеть примерно так:

```
rsync stream tcp nowait root /usr/sbin/tcpd /usr/bin/rsyncd --daemon
```

Когда используется пакет TCP-оболочек, можно сконфигурировать программу `tcpd` таким образом, чтобы блокировать доступ со стороны всех хостов, за исключением того, который будет распространять системные файлы. Аналогичную установку можно сделать и в файле `rsyncd.conf`: никогда не помешает воздвигнуть несколько бастионов.

После того как утилита `rsync` активизирована, следует модифицировать ряд конфигурационных файлов для настройки сервера `rsync`. Основной из них — это `/etc/rsyncd.conf`, содержащий глобальные конфигурационные параметры и описание набора "модулей", каждый из которых представляет собой дерево каталогов для экспорта или импорта. Разумная конфигурация модуля, в который можно копировать файлы (т.е. который будет принимать запросы на пересылку файлов от подключенного клиента), выглядит примерно так:

```
# Название "sysfiles" для данного модуля выбрано произвольно.
[sysfiles]
# Это каталог, в который будут помещаться рассылаемые файлы.
# Может быть просто '/'.
path = /etc
# Это файл, содержащий пары значений имя_пользователя/пароль
# для аутентификации модуля.
secrets file = /etc/rsyncd.secrets
# Модуль должен быть доступен только для чтения в случае
# рассылки по запросу.
read only = false
# Идентификаторы пользователя и группы для операций рассылки.
uid = root
gid = root
# Список хостов, которым разрешено подключаться.
hosts allow = главный_сервер_рассылки
```

Существует множество других опций, но установки по умолчанию вполне приемлемы. В данной конфигурации все операции локализованы в каталоге `/etc`, а доступ разрешен только указанному хосту. С точки зрения пользователя или клиента можно осуществлять рассылку файлов на сервер, указывая в качестве пункта назначения выражение *`хост`*`::sysfile`, которое соответствует описанному выше модулю. Если утилита `rsync` должна работать в режиме рассылки по запросу (файлы загружаются с центрального сервера `rsync`), то показанная конфигурация тоже подойдет (разве что придется включить режим "только чтение").

Наконец, необходимо настроить файл `rsyncd.secrets`. Он обычно хранится в каталоге `/etc` (это может быть любой другой каталог) и содержит пароли, с помощью которых клиенты аутентифицируют себя, например:

```
root:пароль
```

В общем случае пароли, используемые утилитой `rsync`, должны отличаться от реальных паролей. Поскольку пароли отображаются в незашифрованном виде, файл `rsyncd.secrets` должен быть доступен для чтения только суперпользователю.

Рассылка файлов по запросу

Реализовать систему рассылки файлов по запросу можно несколькими способами. Самый простой из них заключается в том, чтобы делать системные файлы доступными на центральном FTP- или Web-сервере [3] и заставлять клиентов автоматически загружать их по мере необходимости. Раньше администраторам приходилось писать для этого собственные утилиты (например, сценарии системы `expect`), но сейчас уже существуют стандартные средства.

Наиболее популярная утилита, входящая в состав большинства дистрибутивов Linux, называется `wget`. Это довольно простая небольшая программа, которая извлекает содержимое по указанному URL-адресу (с использованием протокола FTP либо HTTP). Например, для загрузки файла средствами FTP введите такую команду:

```
wget ftp://пользователь:пароль@хост/путь/файл
```

Указанный файл будет помещен в текущий каталог.

В случае протокола FTP единственной альтернативой является команда `ncftp`, которая тоже есть в большинстве дистрибутивов. Она представляет собой улучшенный вариант FTP-клиента и допускает написание простейших сценариев.

Некоторые организации осуществляют рассылку файлов путем публикации их в сетевых файловых системах типа NFS. С точки зрения автоматизации это, очевидно, простейший способ. Все, что в данном случае необходимо, — это команда `cp`, по крайней мере, теоретически. Но на практике наверняка потребуются еще какие-нибудь средства защиты и проверки целостности файлов. Рассылать важные системные файлы через NFS небезопасно, зато просто и надежно. Вы всегда сможете зашифровать данные, чтобы снизить вероятность их перехвата недоброжелателем.

Остается еще вариант, связанный с использованием утилиты `rsync` (о ней говорилось в предыдущем разделе). Если на центральном хосте рассылки выполняется сервер `rsync`, то клиенты могут загружать файлы с помощью данной утилиты. Это сложнее, чем в случае FTP, зато расширяются функциональные возможности.

Какая бы система ни применялась, следите за тем, чтобы на сервер ложилась посильная нагрузка. Если множество клиентов попытается обратиться к серверу по сети одновременно (например, у каждого из них операция обновления запланирована демоном `cron` на одно и то же время), сервер может, хоть и не преднамеренно, подвергнуться атаке вида “отказ в обслуживании”. Это особенно касается крупных систем, где необходимо обеспечить смещение таких операций по времени или же их запуск в случайном порядке.

[3] Следует иметь в виду, что передача данных по протоколам HTTP и FTP осуществляется посредством открытого текста. Если вопрос защиты данных является крайне важным, можно воспользоваться протоколами HTTPS и SFTP, соответственно.

Проще всего распределять задания демона `cron` с помощью Perl-сценария следующего вида:

```
#!/usr/bin/perl
sleep rand() * 600; # пауза от 0 до 600 секунд (т.е. 10 минут)
system(copy_files_down);
```

17.4. NIS: СЕТЕВАЯ ИНФОРМАЦИОННАЯ СЛУЖБА

Административная СУБД NIS (Network Information Service — сетевая информационная служба), выпущенная компанией Sun в 80-х гг., была первой СУБД такого рода. Сначала она называлась Sun Yellow Pages (желтые страницы Sun), но по причинам правового характера ее пришлось переименовать. Команды NIS до сих пор начинаются с префикса `yp`, поскольку имя, данное при рождении, забыть трудно. СУБД NIS была с воодушевлением принята поставщиками UNIX-систем и поддерживается во всех дистрибутивах Linux.

Единицей совместного использования в NIS является запись, а не файл. Запись обычно соответствует одной строке конфигурационного файла. Главный сервер хранит официальные копии системных файлов, которые находятся в исходных каталогах и имеют текстовый формат. Серверный процесс делает эти файлы доступными по сети. Сервер и его клиенты образуют домен NIS[4].

Для повышения эффективности поиска текстовые файлы преобразуются в файлы базы данных с помощью хеширующих функций библиотеки Berkley DB. После редактирования файлов на главном сервере необходимо попросить NIS преобразовать их в хешированный формат с помощью программы `make`. Полученный файл называется *картой* (map).

С каждой записью может быть связан всего один ключ, поэтому системные файлы иногда приходится транслировать в несколько карт NIS. Например, файл `/etc/passwd` преобразуется в две карты: `passwd.byname` и `passwd.byuid`. Первая служит для отбора записей по имени пользователя, а вторая — для поиска по идентификатору. Любую из них можно использовать для выборки всех записей файла `passwd`, но хеширующие функции не сохраняют порядок записей, поэтому нельзя воссоздать точную копию исходного файла (если только он не хранился в отсортированном виде).

СУБД NIS позволяет реплицировать сетевые карты среди группы подчиненных серверов. Это дает возможность ослабить нагрузку на главный сервер и поддерживать функционирование клиентов даже в том случае, когда некоторые серверы недоступны. Если на главном сервере файл изменился, необходимо разослать соответствующую NIS-карту всем подчиненным серверам, чтобы везде были одинаковые данные. Клиенты не различают главный и подчиненные серверы.

В соответствии с традиционной реализацией NIS, в каждой физической сети должен быть хотя бы один сервер NIS. Для обнаружения серверов клиенты посылают широковещательные IP-пакеты, которые не перенаправляются маршрутизаторами и шлюзами. Чтобы нацелить клиент на конкретный сервер, можно воспользоваться командой `ypset`, однако при первых признаках неполадок клиент попытается обнаружить новый сервер, перейдя в широковещательный режим. Если в клиентской сети нет сервера, это может вызвать зависание клиентского компьютера.

[4] Не следует путать домены NIS с доменами DNS. Это два абсолютно разных понятия, не имеющих между собой ничего общего.

Такая конфигурация вызывает целый ряд проблем, одна из которых — крайне низкая степень безопасности. Злоумышленник может организовать фиктивный сервер NIS, который будет отвечать на широковещательные запросы и либо предоставлять ложные данные, либо осуществлять атаку вида "отказ в обслуживании", позволяя клиентам подключаться к серверу, а затем блокируя их запросы. В наши дни лучше всего предоставлять каждому клиенту явный список легитимных серверов NIS. В результате можно будет не включать серверы в каждую локальную сеть.

В Linux серверы указываются в файле `/etc/yp.conf`. Приведем пример для домена `atrustnis`:

```
domain  atrustnis  server  10.2.2.3
domain  atrustnis  server  10.2.2.4
```

Каждому серверу соответствует отдельная строка. Если сервер выходит из строя, осуществляется переход к следующему серверу в списке. Обратите внимание на то, что указаны IP-адреса серверов. Доменные имена тоже допустимы, но тогда они должны обязательно распознаваться на этапе начальной загрузки (один из способов — перечислить их в файле `/etc/hosts` или распознать с помощью DNS).

Если широковещательный режим необходим, включите такую строку:

```
domain  atrustnis  broadcast
```

Схема работы NIS

Файлы данных NIS хранятся в каталоге `/var/yp`. Далее мы будем называть его "NIS-каталогом". Все NIS-карты хранятся в хешированном виде в подкаталоге NIS-каталога, соответствующем домену NIS. Для каждого поля (ключа), по которому можно осуществить поиск в файле, должна быть создана отдельная карта. Например, в домене `cssuns` DB-карты для файла `/etc/passwd` могут называться так:

```
/var/yp/cssuns/passwd.byname
/var/yp/cssuns/passwd.byuid
```

Поиск в файле `passwd` осуществляется по имени и по идентификатору пользователя, поэтому из него получаются две карты.

Команда `makedbm` создает NIS-карты из обычных файлов. Ее никогда не нужно вызывать непосредственно. Файл `Makefile` в каталоге `/var/yp` сконфигурирован так, чтобы автоматически генерировать все распространенные NIS-карты. После модификации какого-нибудь системного файла перейдите в каталог `/var/yp` и запустите программу `make`. Она сверит время модификации каждого файла со временем модификации соответствующих карт и выполнит команду `makedbm` для каждой карты, которую необходимо перестроить.

Копирование карт с главного сервера на подчиненные осуществляет команда `ypxfr`. Она работает по модели запроса: для того чтобы она импортировала карты, ее нужно запускать на каждом подчиненном сервере. Подчиненные серверы время от времени выполняют команду `ypxfr` для того, чтобы проверить, последние ли версии карт находятся в их распоряжении. С помощью демона `cron` можно управлять периодичностью этих проверок.

Стандартная реализация механизма копирования карт несколько неэффективна, поэтому в Linux существует демон `rpc.ypxfrd`, который можно запустить на главном сервере для ускорения ответов на запросы команды `ypxfr`. Этот демон работает в обход стандартного протокола NIS и просто рассылает копии файлов карт.

Команда `yppush` используется на главном сервере. Она не пересылает никаких данных, а просто заставляет каждый подчиненный сервер выполнить команду `ypxfr`. Команда `yppush` задается в файле `Makefile`, находящемся в NIS-каталоге, и обеспечивает принудительную рассылку откорректированных карт на подчиненные серверы.

Существует специальная карта `ypservers`, которая не соответствует ни одному обычному файлу. Она содержит список всех серверов домена и создается автоматически при конфигурировании домена с помощью команды `ypinit` (о ней речь пойдет ниже). Содержимое этой карты изучается всякий раз, когда главному серверу нужно разослать карты на подчиненные серверы.

После начального конфигурирования единственными активными компонентами системы NIS остаются демоны `ypserv` и `ypbind`. Первый из них работает только на серверах (и главном, и подчиненных); он принимает запросы от клиентов и отвечает на них, осуществляя поиск информации в хешированных файлах карт.

Демон `ypbind` работает на всех компьютерах NIS-домена, включая серверы. Функции библиотеки языка C обращаются к локальному демону `ypbind` всякий раз, когда им нужно ответить на административный запрос (при условии, что это разрешено установками файла `/etc/nsswitch.conf`). Демон `ypbind` находит в соответствующем домене демон `ypserv` и возвращает его адрес библиотечной функции, которая затем обращается непосредственно к серверу. Механизм обработки запроса изображен на рис. 17.1.

Современные версии демона `ypbind` в системах Linux периодически проверяют, работают ли они с наиболее активным сервером в домене NIS. Это является улучшением по сравнению с традиционной реализацией демона, в которой осуществлялась привязка к определенному серверу. Еще одной особенностью, уникальной для Linux, является то, что клиенты могут связывать различные домены NIS с различными картами.

В NIS есть ряд вспомогательных команд, которые предназначены для изучения карт, определения версий карт, используемых каждым сервером, и управления привязкой клиентов к серверам. Полный перечень команд и демонов NIS приведен в табл. 17.2.

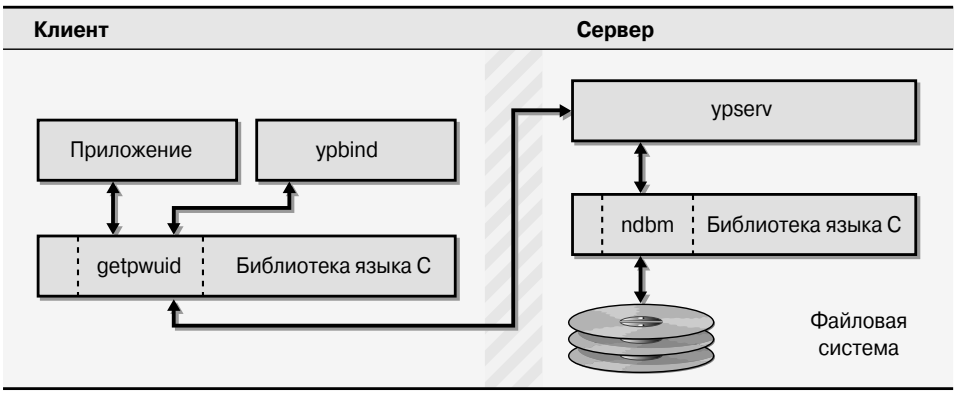


Рис. 17.1. Процедура обработки запросов в NIS

Таблица 17.2. Команды и демоны NIS

Утилита	Описание
`ypserv`	Демон сервера NIS, запускается на этапе начальной загрузки
`ypbind`	Демон клиента NIS, запускается на этапе начальной загрузки
`domainname`	Задает домен NIS, в который входит компьютер (выполняется на этапе начальной загрузки)
`ypxfr`	Загружает текущую версию карты с главного сервера
`ypxfrd`	Обслуживает запросы, поступающие от команды `ypxfr` (работает на главном сервере)
`yppush`	Заставляет подчиненные серверы обновить свои версии карты
`makedbm`	Создает хешированную карту из обычного файла
`ypmake`	Обновляет хешированные карты для тех файлов, которые изменились
`ypinit`	Конфигурирует компьютер как главный или подчиненный сервер
`ypset`	Заставляет демон `ypbind` установить соединение с конкретным сервером[a]
`ypwhich`	Определяет, с каким сервером работает текущий компьютер
`yppoll`	Определяет, какую версию карты использует сервер
`ypcat`	Отображает значения, содержащиеся в NIS-карте
`ypmatch`	Отображает элементы карты, соответствующие заданному ключу
`yppasswd`	Изменяет пароль на главном сервере NIS
`ypchfn`	Изменяет содержимое поля GECOS на главном сервере NIS
`ypchsh`	Меняет регистрационный интерпретатор команд на главном сервере NIS
`yppasswdd`	Сервер для команд `yppasswd`, `ypchsh` и `ypchfn`

[a] Нужно отдельно включать с помощью команды `ypbind -ypsetme` или `ypbind -ypset` (второй вариант опаснее).

Преимущества и недостатки NIS

У СУБД NIS есть одно достоинство: ее могут понять простые смертные. Она работает аналогично службе копирования файлов. В большинстве случаев администраторам не нужно ничего знать о внутренних форматах данных NIS. Администрирование осуществляется с использованием привычных текстовых файлов, нужно лишь дополнительно изучить всего одну-две новые процедуры.

Поскольку механизма объединения доменов в СУБД NIS нет, она годится для управления крупной сетью только при условии, что для каждого входящего в сеть компьютера будет принята одна и та же конфигурация. Можно разделить такую сеть на несколько NIS-доменов, но тогда каждый из них должен администрироваться отдельно. Даже если в крупной сети принята единая конфигурация, ограничение на число подчиненных серверов означает, что в таких организациях существует какой-то другой механизм для синхронизации серверов NIS. Часто имеются резервные базы данных, и серверы NIS черпают данные из этого центрального источника.

Если подчиненный сервер в момент изменения карты не работает или недоступен, хранящаяся на нем копия откорректирована не будет. Подчиненные серверы должны периодически опрашивать главный сервер, чтобы иметь самые последние версии всех карт. Хотя в NIS имеются базовые средства, позволяющие это делать, требуемую схе-

му опроса приходится реализовывать с помощью демона cron. Но даже в этом случае существует вероятность того, что две разные версии одной карты в течение некоторого времени будут обслуживаться одновременно, при этом клиенты, получая доступ к различным подчиненным серверам, будут попеременно видеть то одну, то другую версию.

📖 Более подробно демон cron рассматривался в главе 8.

Система NIS небезопасна. Особенно неприятен широковещательный режим. Любой хост сети может выступить в качестве сервера для того или иного домена и распространить среди NIS-клиентов ложные административные данные. Чтобы избежать этого, следует явно указать каждому клиенту доступные ему серверы NIS.

Можно ограничить круг компьютеров, которым разрешено читать карты с сервера, перечислив их адреса в файле /etc/ypserv.conf, но и такой способ нельзя считать полностью безопасным. Можно также повысить безопасность системы, распространяя файл скрытых паролей по другим каналам (rdist или rsync); не рекомендуется использовать NIS для управления скрытыми паролями.

В старых версиях Linux СУБД NIS содержала известные "бреши". Если используется одна из таких старых систем, обновите ее, прежде чем запускать NIS.

Задание приоритетов для источников административной информации

Информацию о конфигурации можно распространять несколькими способами. Любая система способна обрабатывать обычные файлы и осуществлять поиск имен компьютеров и IP-адресов в DNS. В большинстве систем поддерживается также СУБД NIS. Поскольку для каждого элемента информации может существовать несколько потенциальных источников, в Linux предусмотрен способ задания источников и порядка их опроса.

В первоначальной (до появления Linux) реализации NIS некоторые конфигурационные файлы (в частности, /etc/passwd и /etc/group) должны были "втягивать" в себя содержимое соответствующих карт NIS. Для этого в сами файлы вносились особые обозначения. Одинокий символ "+" в начале строки означал включение всей карты NIS, выражение +@*сетевая_группа* задавало включение записей, относящихся только к указанной группе, а выражение +*имя* соответствовало отдельной записи.

У этого подхода никогда не было много сторонников, поэтому в большинстве систем был введен центральный конфигурационный файл /etc/nsswitch.conf, позволяющий для каждого типа административной информации указывать явный путь поиска. Исходное поведение NIS можно эмулировать в режиме совместимости, но это едва ли понадобится делать в только что сконфигурированной сети (к сожалению, в большинстве дистрибутивов режим эмуляции включен по умолчанию).

Типичный файл nsswitch.conf выглядит следующим образом:

```
passwd:  files nis
hosts:   files dns
group:   files
...
```

Каждая строка соответствует отдельному типу информации (обычно это эквивалент одного текстового файла). Обычные источники таковы: nis, nisplus, files, dns и

compat. Им соответствуют (в порядке перечисления) NIS, NIS+[5], обычные текстовые файлы (без учета специальных обозначений вроде "+"), DNS и обычные файлы системы NIS (с учетом специальных обозначений). DNS предоставляет информацию только об хостах и сетях.

Поддержка каждого типа информации обеспечивается благодаря общей библиотеке (/lib/libnss*), поэтому дистрибутивы могут несколько отличаться тем, какие источники они поддерживают. Некоторые дистрибутивы обеспечивают идеальную поддержку LDAP (см. раздел 17.5) и/или Hesiod, службу каталогов, основанную на DNS. Другим распространенным источником, поддерживаемым в Linux (к сожалению, он плохо документирован), является СУБД Berkeley DB (db), которая читает хешированную версию карты из каталога /var/db (например, /var/db/passwd.db). Если обычные текстовые файлы слишком велики, применение их хешированных версий позволяет существенно повысить скорость поиска информации в этих файлах.

Источники просматриваются слева направо, пока один из них не выдаст ответ на запрос. В показанном выше примере функция gethostbyname сначала проверит файл /etc/hosts и, если требуемый хост там не указан, обратится к DNS. В процессе обработки запросов, касающихся Linux-групп, будет проверяться только файл /etc/group.

В случае необходимости можно явно указать, как следует поступать при получении отрицательного ответа на запрос от того или иного источника. Соответствующее условие берется в квадратные скобки. Например, строка

```
hosts:  dns [NOTFOUND=return] nis
```

заставляет получать информацию только от DNS, если эта служба доступна. Получение отрицательного ответа от сервера имен приведет к немедленному завершению запроса (с выдачей кода ошибки), и обращения к NIS не произойдет. В то же время, СУБД NIS будет задействована, если все серверы имен окажутся недоступными. В табл. 17.3 перечислены различные условия проверки ошибок. Каждое из них может быть задано равным return или continue, что означает, соответственно, прерывание запроса или переход к следующему источнику.

Таблица 17.3. Условия проверки ошибок в файле /etc/nsswitch.conf

Условие	Смысл
UNAVAIL	Источник не существует или недоступен
NOTFOUND	Источник существует, но не может ответить на запрос
TRYAGAIN	Источник существует, но занят
SUCCESS	Источник смог ответить на запрос

Во все дистрибутивы Linux входит файл nsswitch.conf, который подходит для изолированного компьютера, где нет NIS. Источником всех данных служат обычные файлы, за исключением информации об хостах (сначала опрашиваются файлы, потом — DNS). В наших демонстрационных дистрибутивах для файлов passwd и group задан источник compat, что, скорее всего, нежелательно. Если вы действительно используете NIS, просто вставьте этот источник в файл nsswitch.conf.

[5] Неудачный "наследник" исходной версии NIS уже не поддерживается фирмой Sun, хотя в некоторых системах его все еще можно встретить.

В Debian и Ubuntu для элементов `protocols`, `services`, `ethers` и `rpc` основным источником является `db`, следующим — `files`. Это довольно странно, потому что в Debian и Ubuntu нет ни каталога `/var/db`, ни какого бы то ни было механизма для его обслуживания. Очевидно, лучше вообще удалить источник `db`.

Сетевые группы

Вместе с СУБД NIS появилось новое понятие, ставшее вскоре весьма популярным: так называемые *сетевые группы*. Это именованные совокупности пользователей, компьютеров и сетей, на которые делается ссылка в системных файлах. Сетевые группы определяются в файле `/etc/netgroup` и становятся известными клиентам сети через NIS-карту.

Формат записи файла `netgroup` таков:

имя_группы список_членов

Имена членов группы разделяются пробелами или символами табуляции. В свою очередь, член группы может представлять собой имя сетевой группы или триплет следующего вида:

(*имя_компьютера*, *имя_пользователя*, *домен_NIS*)

Каждое пустое поле в триплете воспринимается как метасимвол. Так, элемент `(boulder,,)` ссылается на всех пользователей во всех доменах на компьютере boulder (или на сам компьютер boulder, в зависимости от контекста, в котором используется имя данной сетевой группы). Знак "-" в том или ином поле означает отрицание. Например, элемент `(boulder,-,)` указывает на компьютер boulder и отсутствие пользователей. Определения групп могут быть вложенными.

Вот простой пример файла `/etc/netgroup`:

```
bobcats         (snake,,) (headrest,,)
servers         (anchor,,) (moet,,) (piper,,) (kirk,,)
anchorclients   (xx,,) (watneys,,) (molson,,)
beers           (anchor,,) (anchor-gateway,,) anchorclients
allhosts        beers bobcats servers
```

Все эти сетевые группы определены в виде совокупностей компьютеров, что весьма типично для реальных условий.

Сетевые группы можно использовать в различных системных файлах, определяющих права доступа. Чаще всего с их помощью настраиваются экспортируемые каталоги NFS. Например, в файле `/etc/exports` сетевые группы указывают на то, какие компьютеры имеют право монтировать файловые системы. Это очень удобно, если файловая система экспортируется на множество компьютеров и необходимо указывать полностью определенные доменные имена, потому что длина записи в файле `exports` часто ограничена 1024 символами.

Об NFS рассказывалось в главе 16.

Сетевые группы — прекрасная идея. Их использование помогает упростить системные файлы и сделать их более понятными. Они также дают возможность ввести дополнительный уровень абстракции, который позволяет менять статус пользователя или компьютера путем редактирования одного файла, а не пятнадцати.

Создание домена NIS

Систему NIS следует инициализировать на главном сервере, на подчиненных серверах и на каждом клиенте. Это осуществляется в два этапа. Сначала необходимо выполнить команду `ypinit` на каждом сервере. Затем на каждом компьютере домена следует задать доменное имя в файле `/etc/domainname` или в одном из сценариев запуска системы и настроить файл `/etc/nsswitch.conf` на импорт данных NIS.

Конфигурирование серверов NIS

Сторону сервера NIS нужно инсталлировать в виде отдельного необязательного пакета `ypserv`. В Debian и Ubuntu все обстоит несколько иначе; их пакет `nis` включает как сторону сервера, так и сторону клиента.

Для инициализации главного и подчиненных серверов домена предназначена команда `ypinit`. На главном сервере последовательность команд будет такой:

```
# cd /var/yp                    /* переход в NIS-каталог */
# domainname домен              /* задание имени нового домена */
# /usr/lib/yp/ypinit -m         /* инициализация главного сервера */
# ypserv                        /* запуск сервера NIS */
```

Флаг `-m` команды `ypinit` указывает на то, что конфигурируется главный сервер. Команда сама предложит ввести список подчиненных серверов. После запуска главного сервера необходимо проинициализировать каждый подчиненный сервер, для чего выполняется команда `ypinit` с флагом `-s`:

```
# cd /var/yp
# ypinit -s главный_сервер
# ypserv
```

Команда `ypinit -s` создает локальную копию текущих данных главного сервера. Наличия файлов данных для какого-либо домена достаточно для того, чтобы демон `ypserv` узнал о необходимости обслуживать этот домен.

На каждом подчиненном сервере необходимо задать `crontab`-элементы, которые запрашивают извлечение свежих копий всех карт с главного сервера. Команда `ypxfr` *карта* (где *карта* — это имя наподобие `passwd.byuid`) загрузит указанную карту с главного сервера. Эта команда выполняется один раз для каждой карты. Карты, как правило, изменяются не одновременно, и если пропускная способность сети ограничена, то имеет смысл одни карты пересылать чаще, чем другие. В большинстве случаев вполне достаточно пересылать все карты один-два раза в день (лучше поздно вечером). Следующий сценарий организует пересылку всех карт:

```
#!/bin/sh
mydomain = `/bin/domainname`
cd /var/yp/$mydomain        # NIS-каталог
for map in `/bin/ls`; do
   /usr/lib/yp/ypxfr $map
done
```

Кроме того, в каталоге `/usr/lib/yp` существуют готовые сценарии (`ypxfr_1perday`, `ypxfr_2perday` и `ypxfr_1perhour`), которые организуют пересылку NIS-карт с различной периодичностью.

Если нужно предоставить пользователям возможность изменять свои пароли с помощью команды `yppasswd`, запустите демон `yppasswdd` на главном сервере NIS. Linux-

версия этого демона известна своей нестабильностью, поэтому при возникновении проблем с командой `yppasswd` убедитесь в том, что демон работает.

Настройка параметров управления доступом в файле `/etc/ypserv.conf`

Настроить параметры для Linux-версии демона `ypserv` можно в файле `/etc/ypserv.conf`; однако определить можно только несколько параметров, а для большинства сайтов не нужно изменять их значения, принятые по умолчанию.

Нужно сказать, что `ypserv` ищет в файле `ypserv.conf` инструкции, в которых говорится об управлении доступом к данным NIS. Вместо того чтобы просто посылать ответы на каждый поступающий запрос, как это делается в традиционных реализациях, демон `ypserv` для Linux проверяет, что определено для данного поступившего запроса в списке управления доступом. Каждая строка в этом списке пишется по следующему шаблону:

```
хост:домен_NIS:карта:защита
```

Здесь `хост`, `домен_NIS` и `карта` определяют некоторую подборку запросов, а параметр `защита` говорит о том, как нужно с ней поступать: `deny` приводит к отклонению запроса, `port` — разрешается запрос, если он был послан с привилегированного сетевого порта (<1024), `none` — запрос разрешается всегда. Ниже приведен пример конфигурации.

```
128.138.24.0/255.255.252.0:atrustnis:*:none
*:*:passwd.byuid:deny
*:*:passwd.byname:deny
128.138.:atrustnis:*:port
*:*:*:deny
```

В полях `хост`, `домен_NIS` и `карта` можно использовать знак звездочки, чтобы сопоставлять любое значение, хотя частичные совпадения не допускаются. (Нельзя, например, использовать запись `passwd.*`, чтобы сопоставить все карты, полученные из файла `/etc/passwd`.) Управляющие строки проверяются до тех пор, пока не будет найдена совпадающая строка. Если ни одна строка не совпадет с указанной записью, то на запрос будет дан ответ, используемый по умолчанию.

📖 Сетевые маски CIDR рассматривались в разделе 12.4.

Параметр `хост` может включать сетевую маску, как в первой строке, однако `ypserv` не понимает более обобщенного обозначения CIDR. Как видно в четвертой строке, можно даже пропустить завершающие компоненты IP-адреса, чтобы `ypserv` вместо них поставил нули и создал аналогичную сетевую маску.

Перечисленные правила разрешают доступ к любому хосту в одной из сетей 128.138.24/22. Хосты внутри диапазона 128.138 могут обращаться ко всем картам в компьютере `atrustins`, кроме тех, которые были взяты из файла `/etc/passwd`, при условии, что запрос был сделан на привилегированном порту. Любой другой доступ запрещается.

Следует всегда помнить о том, что такой тип управления доступом является временным решением. Он может запретить доступ людям не из вашей организации, но он не является надежным средством защиты от опытного злоумышленника.

Более старый механизм защиты — использование файла /var/yp/securenets — тоже поддерживается "ради истории". В новых конфигурациях следует использовать файл ypserv.conf.

Конфигурирование клиентов NIS

После конфигурирования серверов второй этап заключается в информировании всех клиентов о том, что они стали членами нового домена. Серверы домена, как правило, являются одновременно его клиентами.

Команда domainname задает NIS-домен, к которому принадлежит компьютер. Обычно она вызывается на этапе начальной загрузки из какого-нибудь сценария запуска системы; детали этого процесса зависят от дистрибутива.

О сценариях запуска системы рассказывалось в главе 2.

Каждый клиент должен иметь хотя бы минимальные личные версии файлов passwd, group и hosts. Первые два необходимы для того, чтобы пользователь root мог войти в систему при отсутствии сервера NIS. Они должны содержать описание стандартных системных учетных записей и групп: root, bin, daemon и др. Файл hosts (или служба DNS) нужен как источник информации на этапе начальной загрузки, пока СУБД NIS не начала работать.

О тонкостях NIS в разных дистрибутивах

В системах Fedora и RHL имя домена NIS задается в переменной NISDOMAIN файла /etc/sysconfig/network. Сторона сервера NIS инсталлируется в виде отдельного пакета ypserv. Запуск демонов ypbind, ypserv и yppasswdd включается и отключается с помощью команды chkconfig, например:

```
# chkconfig ypbind on
```

В SUSE доменное имя NIS задается во время загрузки системы из файла /etc/domainname. Сторона сервера NIS инсталлируется в виде отдельного пакета ypserv. С помощью утилиты chkconfig можно автоматически запускать демон ypserv и/или ypbind во время загрузки системы. В файле /etc/sysconfig/ypbind можно задать опции командной строки демона ypbind. Вы должны или присвоить опции YPBIND_BROADCAST в этом файле значение yes , или инсталлировать файл /etc/yp.conf. В противном случае сценарии запуска проигнорируют демон ypbind.

В Debian и Ubuntu имя домена NIS хранится в файле /etc/defaultdomain. Демон ypbind запускается автоматически, если файл /etc/defaultdomain существует. Чтобы запустить ypserv, нужно в файле /etc/init.d/nis присвоить переменной NISSERVER значение true.

17.5. LDAP: УПРОЩЕННЫЙ ПРОТОКОЛ ДОСТУПА К КАТАЛОГАМ

Организациям, где применяется UNIX и Linux, необходим надежный способ распространения своих административных данных. Но проблема в действительности более глобальна. Как быть с неадминистративными ресурсами, например каталогами элек-

тронной почты? Как контролировать информацию, предоставляемую внешнему миру? Решением, которое устроило бы всех, является унифицированная служба каталогов.

Служба каталогов — это просто база данных, но такая, в которой сделан ряд дополнительных предположений. Любой набор данных, характеристики которых подпадают под эти предположения, становится кандидатом на включение в базу данных. Основные предположения таковы:

- информационные объекты относительно невелики;
- база данных будет реплицироваться и кэшироваться на множестве компьютеров;
- у информации есть атрибуты;
- данные извлекаются часто, но записываются редко;
- операции поиска выполняются очень часто.

Текущая стандартная система, предложенная организацией IETF для этих целей, называется LDAP (Lightweight Directory Access Protocol — упрощенный протокол доступа к каталогам). В спецификациях LDAP говорится не о самой базе данных, а лишь о том, как получить к ней доступ по сети. В то же время, заданы схемы организации данных и осуществления поиска, поэтому подразумевается достаточно четкая модель данных.

Изначально LDAP замышлялся как простой шлюзовой протокол, который позволял бы клиентам TCP/IP взаимодействовать с серверами каталогов X.500, теперь уже устаревшими. Со временем стало очевидно, что стандарт X.500 изжил себя и в UNIX необходима стандартная служба каталогов. Все это привело к тому, что из LDAP получилась совершенно самостоятельная, полноценная система управления каталогами (возможно, буква "L" в названии стоит незаслуженно)[6].

По состоянию на 2006 г. протокол LDAP получил широкое распространение, чему отчасти способствовало принятие его фирмой Microsoft в качестве основы для службы Active Directory. В среде UNIX и Linux стандартной реализацией стал пакет OpenLDAP (`www.openldap.org`).

Структура данных LDAP

Данные протокола LDAP принимают форму списков свойств, которые в мире LDAP называют "записями" (entry). Каждая запись состоит из набора именованных атрибутов (например, "uid", "description") и значений этих атрибутов. Пользователи, работающие в среде Windows, обнаружат, что эта структура напоминает структуру записей в системном реестре. Как и в реестре Windows, отдельный атрибут может иметь несколько различных значений.

В качестве примера рассмотрим обычную (и упрощенную) строку файла `/etc/passwd`, выраженную в виде записи LDAP:

```
uid: ghopper
cn: Grace Hopper
userPassword: {crypt}$1$pZaGA2RL$MPDJoc0afuhHY6yk8HQFp0
loginShell: /bin/bash
uidNumber: 1202
gidNumber: 1202
homeDirectory: /home/ghopper
```

[6] Вследствие сложной истории LDAP во многих книгах можно встретить подробные рассказы о X.500 и OSI. Однако эта история не имеет никакого отношения к современному использованию LDAP. Просто забудьте о ней.

Здесь мы видим простой пример формата LDIF (LDAP Data Interchange Format — формат обмена данными LDAP), который применяется в большинстве инструментов, связанных с LDAP, и в реализациях серверов. Причиной успеха этого формата является то обстоятельство, что LDAP можно без труда преобразовывать в простой текст и обратно.

Записи систематизируются по "отличительным именам" (имя атрибута: `dn` — distinguished name), которые формируют некоторую разновидность пути поиска. Например, запись `dn` для вышеупомянутого пользователя может выглядеть так:

```
dn: uid=ghopper,ou=People,dc=navy,dc=mil
```

Как и в DNS, самый значащий бит ставится справа. Здесь имя DNS `navy.mil` используется для построения верхних уровней иерархии LDAP. Оно разбивается на два компонента домена: `navy` и `mil`, хотя это всего лишь одно из некоторых обычных соглашений.

Каждая запись имеет одно отличительное имя. Поэтому иерархия записей будет выглядеть подобно простому ветвлению без циклов. Кроме того, существуют также и резервы для символических ссылок между записями и для отсылок на другие серверы.

Записи LDAP обычно составляются на основе использования атрибута `objectClass` (объектный класс). Классы объектов определяют атрибуты, которые может содержать запись, причем некоторые из них могут требоваться для проверки достоверности. Каждому атрибуту назначается тип данных. Схема вложения и комбинирования классов объектов ничем не отличается от схемы, принятой в традиционном объектном ориентировании. Верхний уровень дерева класса объектов называется `top`; он определяет только то, что запись должна иметь атрибут `objectClass`.

В табл. 17.4 показаны некоторые обычные атрибуты LDAP, назначение которых с первого раза может быть непонятным.

Таблица 17.4. Некоторые имена атрибутов, которые можно встретить в иерархиях LDAP

Атрибут	Расшифровка	Для чего предназначен
`O`	Organization (организация)	Часто идентифицирует запись верхнего уровня сайта
`Ou`	Organization unit (организационная единица)	Логическое подразделение, например "marketing" (отдел маркетинга)
`Cn`	Common name (обычное имя)	Наиболее естественное имя, с помощью которого можно передать смысл записи
`Dc`	Domain component (компонент домена)	Используется в сайтах, в которых иерархия LDAP построена на основе DNS
`objectClass`	Object class (объектный класс)	Схема, в соответствии с которой формируются атрибуты данной записи

Особенности LDAP

Для уверенной работы с LDAP вам нужно будет приобрести хотя бы небольшой опыт. В отличие от NIS, LDAP сам по себе не решает какую-либо административную проблему. Нет какой-то "главной задачи", которую можно было бы решить исключительно с помощью LDAP; к тому же, на разных сайтах по-разному подходят к необходимости развертывания серверов LDAP. Поэтому прежде чем мы перейдем к особенностям инсталляции и конфигурирования OpenLDAP, мы поговорим о том, в каких случаях целесообразно использовать LDAP в рамках данного сайта.

- LDAP можно использовать в качестве замены NIS. Вы получите много преимуществ, однако главным будет улучшенная защита, улучшенная инструментальная поддержка и более высокая степень гибкости.
- Продолжая тему дальше, LDAP можно использовать для распространения информации о конфигурации, которая выходит за рамки традиционной области NIS. Многие почтовые системы, включая `sendmail` и Postfix, могут узнать много информации о маршрутизации из LDAP, и этим отчасти можно объяснить популярность применения LDAP. Инструменты будут отличаться в зависимости от конфигурации Web-сервера Apache и `autofs`. Вероятно, со временем поддержка LDAP будет все более и более расширяться.
- Доступ к данным LDAP производится с помощью инструментов командной строки, таких как `ldapsearch`. Кроме того, LDAP всесторонне поддерживается языками написания сценариев вроде Perl и Python (посредством использования библиотек). Следовательно, LDAP выгодно использовать для передачи информации о конфигурации локально написанным сценариям и утилитам администрирования.
- Для управления LDAP доступен превосходный Web-инструментарий, например: phpLDAPadmin (`phpldapadmin.sourceforge.com`) и Directory Administrator (`diradmin.open-it.org`). Эти инструменты настолько просты в использовании, что вы сможете приступить к работе с ними, даже не читая руководства.
- LDAP хорошо поддерживается и как служба общедоступных каталогов. Многие ведущие почтовые клиенты, включая те, которые работают на ПК, поддерживают LDAP для доступа к каталогам пользователей. Простой поиск с использованием LDAP поддерживается во многих Web-браузерах посредством использования типа LDAP URL.
- Служба Microsoft Active Directory построена на LDAP, а последняя версия Windows 2003 Server R2 включает расширения (раньше они назывались "Службы для UNIX", а сейчас — "Службы каталогов и защиты Windows для UNIX"), с помощью которых можно выполнять отображение пользователей и групп Linux.

Документация и спецификации LDAP

Нам не известно о каком-либо действительно превосходном источнике предварительной информации об общей архитектуре LDAP. Разовые ответы на часто задаваемые вопросы уже не поддерживаются. В качестве введения в LDAP version 2 мы можем порекомендовать статью *Understanding LDAP*, написанную Хайнцом Йонером (Heinz Johner) и другими авторами. К сожалению, для третьей версии LDAP нет варианта этой статьи, а различия между протоколами столь серьезные, что им впору посвящать отдельную статью.

Сейчас наилучшей является документация *LDAP HOWTO*, посвященная пакету OpenLDAP и Linux. В конце этой главы вы найдете ссылки на пару полезных книг по LDAP.

Существует множество разнообразных документов RFC, посвященных протоколу LDAP, каждый из которых является довольно сложным. Наиболее важные документы перечислены в табл. 17.5.

Таблица 17.5. Документы RFC, посвященные протоколу LDAP

RFC	Название
4519	LDAP: Schema for User Applications
4517	LDAP: Syntaxes and Matching Rules
4516	LDAP: Uniform Resource Locator
4515	LDAP: String Representation of Search Filters
4514	LDAP: String Representation of Distinguished Names
4513	LDAP: Authentication Methods and Security Mechanisms
4512	LDAP: Directory Information Models
4511	LDAP: The Protocol
3672	Subentries in the Lightweight Directory Access Protocol (LDAP)
3112	LDAP Authentication Password Schema
2849	LDAP Data Interchange Format (LDIF)—Technical Specification
2820	Access Control Requirements for LDAP
2307	An Approach for Using LDAP as a Network Information Service

Настройка не является сложной. В первую очередь потребуется создать файл `/etc/openldap/slapd.conf`, скопировав образец, который был инсталлирован вместе с сервером OpenLDAP. Необходимо обратить внимание на следующие строки:

```
database bdb
suffix "dc=mydomain, dc=com"
rootdn "cn=администратор, dc=mydomain, dc=com"
rootpw {crypt}abJnggxhB/yWI
directory /var/lib/ldap
```

По умолчанию выбирается формат базы данных Berkley DB, отлично подходящий для данных, манипуляция которыми будет производиться в системе OpenLDAP. Вы можете использовать разнообразные интерфейсы, включая специальные методы, такие как сценарии, создающие данные "на лету".

Переменная `suffix` задает "базовое имя" LDAP. Это корень вашей части пространства имен LDAP, то, что в DNS называется доменным именем. Этот пример показывает обычное использование доменного имени в качестве базового имени LDAP.

Переменная `rootdn` задает административное имя, а переменная `rootpw` — пароль администратора в стандартном формате UNIX (алгоритм DES). Обратите внимание на то, что доменные компоненты, восходящие к административному имени, также должны быть указаны. Пароль можно либо скопировать из файла `/etc/shadow` (если не используются пароли MD5), либо сгенерировать с помощью простейшего Perl-сценария:

```
perl -e "print crypt('пароль','примесь');"
```

где *примесь* — это произвольная двухсимвольная строка. Из-за присутствия пароля следует убедиться в том, что режим доступа к файлу `slapd.conf` равен 600, а его владельцем является пользователь `root`.

Нужно также отредактировать файл `/etc/openldap/ldap.conf`, чтобы указать сервер по умолчанию и базовое имя для клиентских запросов LDAP. Это несложно: просто задайте аргумент записи `host` равным имени сервера, а в переменную `base` занесите то

же значение, что и в переменной `suffix` файла `slapd.conf` (убедитесь, что обе строки не являются комментариями).

С этого момента можно запускать демон `slapd` без аргументов.

Замена NIS на LDAP

Вам не нужно использовать NIS, чтобы перейти к LDAP, но поскольку NIS определяет стандартный набор общих системных файлов, она служит в качестве полезного ориентира.

📖 О подключаемых модулях аутентификации будет рассказано в конце раздела 20.5.

Файлы, распространяемые посредством NIS, уже "виртуализированы" посредством переключателя службы имен (файл `/etc/nsswitch.conf`), поэтому включить поддержку LDAP на стороне клиента будет несложно. В некоторых дистрибутивах необходимый пакет `nss_ldap` инсталлируется по умолчанию; в противном случае пакет можно найти в качестве опции. Этот пакет включает модуль PAM, с помощью которого можно использовать LDAP с подключаемыми модулями аутентификации в дополнение к переключателю службы имен.

Параметры этого пакета, используемые по умолчанию в LDAP на стороне клиента, задаются в файле `/etc/ldap.conf`, который совместно использует свой формат с файлом `/etc/openldap/ldap.conf`, описанном в предыдущем разделе, но который включает дополнительные опции, специфические для службы имен и контекста PAM. Вы должны будете также отредактировать файл `/etc/nsswitch.conf` на каждом клиенте, чтобы добавить `ldap` в качестве источника для каждого типа данных, который будет приведен к стандарту LDAP. (Изменения в файле `nsswitch.conf` приводят к тому, что библиотека языка C передает запросы библиотеке `libnss_ldap`, которая затем использует информацию `/etc/ldap.conf`, чтобы выяснить, как нужно обращаться с запросами LDAP.)

В документе RFC2307 определено стандартное преобразование из традиционных наборов данных UNIX, к которым относятся файлы `passwd` и `group`, в пространство имен LDAP. Это полезный справочный документ (по крайней мере, с теоретической точки зрения) для системных администраторов, использующих LDAP в качестве замены NIS. На практике все эти спецификации читать гораздо проще компьютерам, чем людям; вам же лучше будет просто просмотреть примеры.

Фирма Padl Software предлагает бесплатную коллекцию сценариев, которые переводят существующие текстовые файлы или карты NIS в LDAP. Эти сценарии можно найти по адресу `www.padl.com/tools.html`. Работать с ними очень просто. Сценарии можно использовать в качестве фильтров для генерации LDIF, а также можно запускать, чтобы загружать данные прямо с сервера. Например, сценарий `migrate_group` преобразовывает строку

```
csstaff:x:2033:evi,matthew,trent
```

взятую из `/etc/group`, в следующий блок LDIF:

```
dn: cn=csstaff,ou=Group,dc=domainname,dc=com
cn: csstaff
objectClass: posixGroup
objectClass: top
userPassword: {crypt}x
gidNumber: 2033
memberuid: evi
memberuid: matthew
memberuid: trent
```

(Обратите внимание на спецификации классов объектов и отличительные имена, которые были опущены в примере `passwd` в начале раздела 17.5.)

После импортирования базы данных вы сможете проверить работу преобразования, запустив утилиту `slapcat`, которая отображает всю базу данных.

LDAP и безопасность

Так уж сложилось, что протокол LDAP использовался чаще всего по принципу телефонного каталога, поэтому отправка данных без шифрования была обычным делом. Как результат, "стандартная" реализация LDAP предоставляет незашифрованный доступ через TCP-порт 389. Однако мы настоятельно советуем использовать незашифрованный протокол LDAP для передачи информации об аутентификации, даже если пароли индивидуально хешируются или зашифровываются.

Как вариант, во многих ситуациях (даже в мире Microsoft) LDAP можно использовать на основе SSL (эта комбинация известна как LDAPS; обычно эта пара работает на TCP-порте 686) как на стороне сервера, так и на стороне клиента. Такой способ доступа является предпочтительным, поскольку он защищает информацию, содержащуюся как в запросе, так и в ответе. Желательно использовать LDAPS везде, где это возможно.

Столь сложная система, как LDAP, с большим числом компонентов, неизбежно должна содержать бреши, ослабляющие ее защиту, не говоря уже о брешах, издавна существующих в операционной системе. Поэтому системным администраторам постоянно приходится быть начеку.

17.6. Рекомендуемая литература

- Malère, Luiz Ernesto Pinheiro. *LDAP Linux HOWTO.* `www.tldp.org/HOWTO/LDAP-HOWTO/`
- Volgmaier, Reinhard. *The ABCs of LDAP: How to Install, Run, and Administer LDAP Services.* Boca Raton, FL: Auerbach Publications, 2004.
- Carter, Gerald. *LDAP System Administration.* Sebastopol, CA: O'Reilly Media, 2003.

17.7. Упражнения

17.1. Почему метод обновления файлов локального компьютера по запросу считается безопаснее метода принудительной рассылки?

17.2. Поясните приведенный ниже фрагмент управляющего файла утилиты `rdist`:

```
LINUX_PASSWD = ( redhatbox debianbox susebox )

passwd:
    ( /etc/passwd ) -> ( ${LINUX_PASSWD} )
    install /etc/passwd.rdist;
    cmdspecial /etc/passwd.rdist "/usr/local/sbin/mkpasswd";
```

☆ 17.3. Объясните разницу между утилитами `rdist` и `rsync`. В каких случаях предпочтительнее использовать ту и другую?

☆ 17.4. Сравните системы NIS и LDAP. В каких ситуациях вы отдадите предпочтение одной из них? Бывают ли ситуации, когда полезны обе?

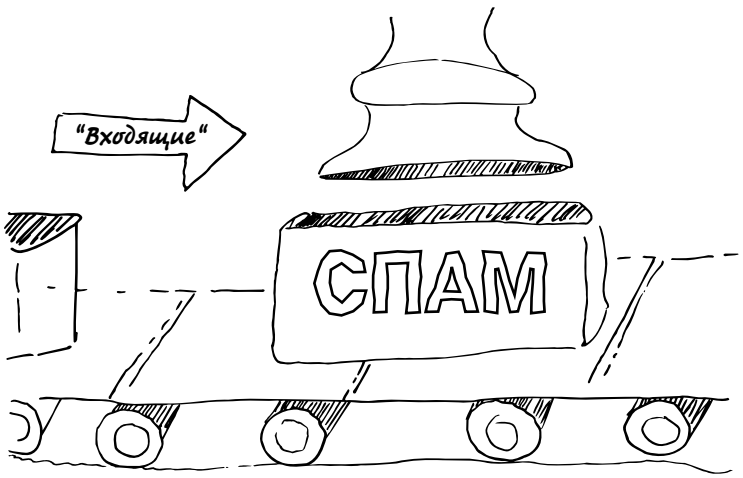
☆ 17.5. Какой метод совместного использования файлов применяется в вашей системе? Какие проблемы безопасности при этом возникают? Предложите альтернативную схему совместного использования файлов и укажите ее достоинства и недостатки.

☆☆☆☆☆ 17.6. Придумайте схему LDAP, которая хранила бы информацию о пользователе, такую как регистрационное имя, пароль, оболочка, авторизированные компьютеры и т.п. Создайте инструмент, с помощью которого можно в интерактивном режиме добавлять новых пользователей в базу данных из файла, содержащего список пользователей. Создайте инструмент, который будет генерировать файлы `passwd`, `group` и `shadow` из базы данных LDAP для компьютеров, работающих в вашем отделе. Разрешите пользователям иметь различные пароли на каждом компьютере. (Чтобы пользоваться каждым компьютером, не всем пользователям нужна обязательная авторизация.) Ваша система `adduser` должна уметь печатать списки регистрационных имен существующих пользователей и печатать пары “регистрационное имя/пароль” для новых пользователей.

Book_LinuxAdmHand.indb 590 17.04.2007 13:55:46

Глава 18

Электронная почта



Сейчас уже сложно представить себе, на что был похож наш мир, когда не существовало электронной почты. В наши дни очень многие люди, независимо от возраста и социального положения, пользуются электронной почтой для общения с близкими и родственниками, сотрудниками и сослуживцами, партнерами, заказчиками и даже с членами кабинета министров. Электронной почтой сейчас можно пользоваться даже в самых укромных уголках земного шара[1].

Электронная почта стала популярной благодаря тому, что ее принцип отправки сообщений очень похож на написание традиционных писем, поэтому понять его могут обыкновенные люди, даже слабо знакомые с компьютерной техникой. Кроме того, электронная почта "просто работает"; если вы знаете адрес электронной почты нужного вам человека, то вам достаточно ввести текст предназначенного ему сообщения и щелкнуть на кнопке "Отправить". Больше ничего делать не нужно! Спустя несколько секунд ваше сообщение окажется в электронном почтовом ящике этого человека, независимо от того, где расположен компьютер, принимающий сообщения — в соседней комнате или на другом полушарии Земли. С точки зрения пользователя, в этом нет ничего сложного.

[1] Даже если вы будете плавать на корабле в пасмурную погоду вдали от берега, вы все равно сможете общаться со своими друзьями и родственниками благодаря повсеместному распространению сети GPRS. Ее покрытие охватывает территории 113 стран.

К сожалению, не всегда все было так просто, и даже в наши дни базовая инфраструктура, благодаря которой работает электронная почта, является сложной для понимания. Существует несколько пакетов программ, которые можно запускать в системе Linux для транспортировки электронной почты и ее управления (в этой главе будет рассмотрено три таких пакета), однако каждую из них вам придется конфигурировать и управлять. Кроме того, вам нужно будет понимать базовые принципы и протоколы, связанные с электронной почтой, чтобы вы не думали, что межплатформенная электронная почта, которая может функционировать между организациями, является божественным даром, и она таинственным образом работает в любой момент времени.

В наши дни существуют альтернативные варианты понимания и администрирования инфраструктуры электронной почты. Многие поставщики услуг предлагают услугу "управляемой" почты: электронная почта обслуживается на серверах центра по обработке данных, принадлежащего поставщику услуг и находящегося где-то очень далеко, а вы ежемесячно или ежегодно платите деньги за право доступа к вашим письмам. Можно пользоваться также и "бесплатными" почтовыми серверами, такими как Yahoo!, Mail, MSN Hotmail и Google Gmail. Если вы являетесь частным лицом и вам нужно пользоваться электронной почтой в личных или деловых целях, то последний вариант может быть очень удобным. Организации, которым принадлежат эти серверы, сами решают вопросы, связанные с местом для хранения почты, управлением серверами, обновлением программ, конфигурацией, фильтрацией спама, защитой и т.п. Взамен того, что вы будете бесплатно пользоваться их почтовыми услугами, они могут вам показывать ту или иную рекламу. Для многих людей этот вариант является очень удобным; если он вам подходит, то вам не придется читать остальной материал, представленный в этой главе.

Тем не менее, вариант с использованием почтовых серверов поставщиков услуг подходит далеко не для всех. Крупные фирмы и организации, которым нужна связь по электронной почте, часто не идут на риск, связанный с размещением почтовых серверов у поставщиков услуг. Такие организации предпочитают иметь собственные системы обработки электронной почты, и это объясняется рядом причин: безопасностью, производительностью и доступностью. Эта глава предназначена людям, работающим в таких организациях.

Огромный объем этой главы свидетельствует о том, что почтовые системы — достаточно сложная тема. Здесь представлены как базовые сведения, так и детали конфигурирования программного обеспечения.

Сначала мы хотели разделить эту главу на четыре поменьше, посвятив их почтовым системам в целом, конфигурации программы `sendmail`, спаму, пакету Exim и Postfix. Однако затем отказались от этой идеи и думаем, что поступили верно. Вместо этого читателям предлагается табл. 18.1, в которой описан каждый раздел главы.

Благодаря такой организации главы изложение материала выглядит более логичным при последовательном чтении, хотя иногда вопросы, касающиеся решения одной конкретной задачи, могут оказаться разнесенными. Читать всю главу имеет смысл почтмейстеру компании среднего размера, но не администратору, собирающемуся настраивать почтовый клиент персонального компьютера.

В табл. 18.2 приведен перечень наиболее распространенных административных задач и номера разделов, где изложена соответствующая информация.

Таблица 18.1. Структура главы

	Раздел	Описание
Основы	1	Системы электронной почты, их компоненты
	2	Адресация, синтаксис адресов, заголовки сообщений
	3	Принципы работы электронной почты, модель клиент/сервер, почтовые каталоги
	4	Псевдонимы, маршрутизация почты, протокол LDAP
	5	Программы для работы со списками рассылки
Конфигурирование `sendmail`	6	Программа `sendmail`: инсталляция, запуск, очередь сообщений
	7	Основы конфигурирования программы `sendmail`, макросы `m4`
	8	Базовые примитивы конфигурации программы `sendmail`
	9	Более сложные примитивы конфигурации программы `sendmail`
	10	Спам, база доступа программы `sendmail`
	11	Примеры конфигураций
	12	Безопасность
	13	Производительность
	14	Статистика, тестирование, отладка
Прочее	15	Пакет Exim как альтернатива программе `sendmail`
	16	Postfix, еще одна альтернатива программе `sendmail`
	17	Дополнительные источники информации

Таблица 18.2. Список задач с указанием соответствующих разделов

Задача	Номера разделов
Обновление программы `sendmail`	6, 7
Начальное конфигурирование программы `sendmail`	3, 6, 7, 8, 9, 10, 11, 12
Изменение конфигурационного файла	7
Проектирование почтовой системы организации	3, 4, 6, 7, 8, 9, 10, 11
Борьба со спамом	10
Проверка безопасности	12
Настройка ПК на прием почты	1, 3
Создание списка рассылки	5
Настройка производительности	3, 9, 13
Виртуальный хостинг	9
Использование пакета Exim вместо `sendmail`	15
Использование Postfix вместо `sendmail`	16

В этой главе в основном рассматривается конфигурация `sendmail` — стандартной программы, которая анализирует и маршрутизирует электронную почту. Эту программу написал Эрик Оллман (Eric Allman) из Калифорнийского университета в Беркли. Существуют три основные версии программы: 5, IDA и 8. Полностью переработанная версия, Sendmail X, недавно была выпущена в виде бета-версии, хотя для использования в организациях она все равно пока что не годится. (Со слов знающих людей, может случиться так, что она вообще никогда не заменит версию 8.) Версии 5 и IDA в настоящее время почти не используются. Ниже мы опишем версию 8 (точнее, 8.13).

Коммерческие версии программы `sendmail` разрабатываются компанией Sendmail, Inc. Эта же компания осуществляет поддержку и свободно распространяемой версии с открытым исходным кодом. Коммерческие версии включают графическую утилиту настройки, централизованные средства мониторинга и составления отчетов (что особенно удобно в крупных почтовых системах).

18.1. Системы электронной почты

Теоретически система электронной почты состоит из четырех компонентов:

- *пользовательского агента*, который дает пользователям возможность читать и составлять сообщения;
- *транспортного агента*, который пересылает сообщения с одного компьютера на другой;
- *агента доставки*, который помещает сообщения в локальное хранилище[2];
- необязательного *агента доступа*, который связывает пользовательский агент с хранилищем сообщений (например, посредством протокола IMAP или POP).

В некоторых организациях используется также *агент подачи почты*, который работает по протоколу SMTP (Simple Mail Transfer Protocol — простой протокол передачи электронной почты) и выполняет часть функций транспортного агента. Схема взаимодействия всех указанных компонентов представлена на рис. 18.1.

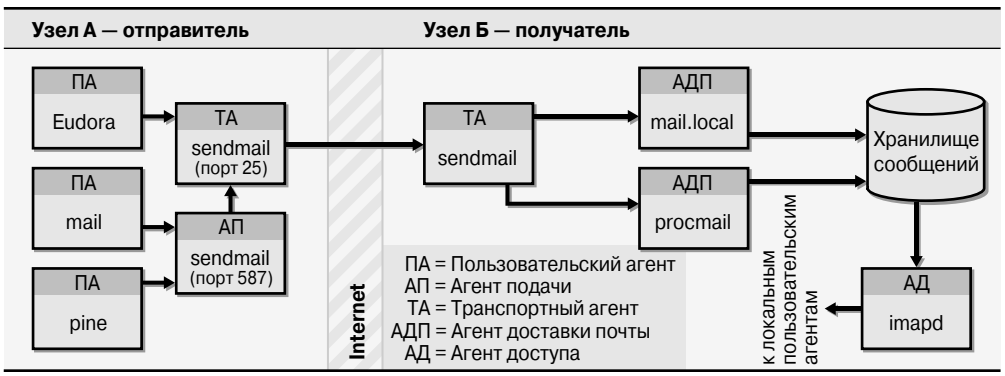


Рис. 18.1. Компоненты почтовой системы

[2] Иногда это почтовые ящики пользователей, иногда — база данных.

Пользовательские агенты

Пользовательский агент применяется для чтения и составления электронных сообщений. Первоначально сообщения могли содержать только простой текст, однако благодаря стандарту MIME (Multipurpose Internet Mail Extensions — многоцелевые расширения электронной почты в сети Internet) появилась возможность включать в них форматированный текст и присоединять разные файлы (в том числе вирусы). Стандарт MIME поддерживается большинством пользовательских агентов. Поскольку он не влияет на процесс адресации и доставки почты, мы не будем его рассматривать.

Одной из задач пользовательского агента является обеспечение того, чтобы текст внутри сообщения, который может быть истолкован почтовой системой неправильно, был "защищен". Примером является строка "From", которая служит разделителем сообщений.

Самым первым пользовательским агентом была утилита `/bin/mail`. Она до сих пор используется для чтения текстовых электронных сообщений из строки приглашения интерпретатора команд. Лучше это или хуже, но эпоха текстовой электронной почты закончилась, поэтому для большинства пользователей текстовые пользовательские агенты больше не представляют практического интереса. Графические пользовательские интерфейсы позволяют производить доступ к сообщениям по типу "навел и щелкнул", и позволяют присоединять к пересылаемым сообщениям файлы (файлы изображений, документы Microsoft Word, электронные таблицы и многое другое).

Как видно на рис. 18.1, одной из примечательных особенностей является то, что пользовательский агент не обязательно должен работать в такой же системе, или даже на такой же платформе, как и остальная часть системы электронной почты. Пользователи могут запускать один из многих пользовательских агентов, входящих в состав дистрибутива Linux, во время регистрации в системе, а также получать доступ к своей электронной корреспонденции посредством протоколов агентов доступа, таких как IMAP или POP, из компьютеров, работающих под управлением Windows. Между прочим, эта конфигурация на сегодняшний день является наиболее распространенной. И кто говорит, что Linux и Windows не могут существовать вместе?

В настоящее время можно воспользоваться несколькими пользовательскими агентами. Вот перечень наиболее популярных из них.

- `/bin/mail` в Red Hat — это BSD-версия исходной UNIX-команды `mail`; в SUSE, Debian и Ubuntu утилита называется `/usr/bin/mail`[3]; Этот пользовательский агент работает только с текстом и требует хранить почту локально.
- Mozilla Thunderbird для Linux, Windows и Mac OS.
- Evolution (или Novell Evolution; первоначально он назывался Ximian Evolution) для Linux, Windows и Mac OS.
- `pine` — разработан в университете штата Вашингтон (`www.washington.edu/pine`);
- Eudora — разработан компанией Qualcomm для систем Macintosh и Windows;
- Outlook — разработан компанией Microsoft для Windows-систем.

[3] В других системах этот агент иногда называется `Mail` или `mailx`, поэтому в Red Hat есть ссылка `Mail`, а в SUSE, Debian и Ubuntu — ссылки `Mail` и `mailx`.

Транспортные агенты

Задача транспортного агента — принимать почту от пользовательского агента, интерпретировать адреса получателей и переправлять почту на соответствующие компьютеры для последующей доставки. Кроме того, большинство транспортных агентов функционирует в качестве агентов подачи почты, осуществляя первоначальную вставку новых сообщений в почтовую систему. Транспортные агенты работают по протоколу SMTP (Simple Mail Transport Protocol — простой протокол передачи электронной почты), который определен в документе RFC2821 (первоначальная версия — в документе RFC821), или по протоколу ESMTP (Extended SMTP), описанному в документах RFC1869, 1870, 1891 и 1985.

Для UNIX и Linux разработано несколько транспортных агентов (PMDF, Postfix, `smail`, `qmail`, Exim, `zmailer` и др.), но самым мощным, гибким и потому самым распространенным является `sendmail`. Исследование[4] почтовых систем, проведенное в 2001 г., выявило, что на долю программы `sendmail` приходится 60% доменов, Exim — 8%, Microsoft Exchange Server — 4% и Postfix — 2%. Остальные программы (их около пятидесяти) имеют очень узкий круг пользователей.

Системы Red Hat, Fedora и SUSE поставляются с уже инсталлированной программой `sendmail`. В Debian тоже есть `sendmail`, но если присмотреться повнимательнее, то окажется, что `sendmail` — это всего лишь ссылка на программу Exim. Ее разработчики потрудились над тем, чтобы программа понимала флаги командной строки `sendmail`. Таким образом, даже если пользовательский агент делает явный вызов `sendmail`, это еще ни о чем не говорит. Система Ubuntu поставляется вместе с программой Exim, которая используется в ней по умолчанию.

Агенты доставки

Агент доставки отвечает за прием почты от транспортного агента и ее передачу соответствующим получателям. Почта может доставляться конкретному пользователю, в список рассылки, в файл и даже в программу.

Для обслуживания получателя каждого типа может понадобиться отдельный агент. Например, программа `/bin/mail` служит агентом доставки для локальных пользователей. Программа `/bin/sh` — агент доставки почты, которая направляется в файл или программу. Последние версии программы `sendmail` осуществляют доставку почты при помощи более надежных локальных агентов доставки, которые называются `mail.local` и `smrsh`. В качестве агента может также использоваться утилита `procmail` (описывается в разделе 18.9), которая доступна на Web-сайте `www.procmail.org`. Точно так же, если в качестве агента доступа запустить программу `imapd`, разработанную компанией Cyrus, то у нее будет свой собственный локальный агент доставки.

Хранилища сообщений

Хранилище сообщений представляет собой место на локальном компьютере, где располагается электронная почта. Традиционно для этого используется каталог `/var/spool/mail` или `/var/mail`. Почта хранится в файлах, которые называются в соответствии с регистрационными именами пользователей. Однако так выглядит обычное хранилище. Провайдеры, у которых тысячи или миллионы клиентов, ищут другие технологии организации хранилищ (в основном это базы данных).

[4] Частное исследование хоста Matrix.net для компании Sendmail, Inc.

В системах, где применяется каталог /var/spool/mail или /var/mail, он создается в процессе инсталляции операционной системы. Режим доступа к каталогу должен быть равен 775, имя группы — mail[5]. Но есть исключение: если в качестве локального агента доставки используется программа mail.local, режим доступа может быть 755. Настройки наших демонстрационных дистрибутивов несколько отличаются друг от друга:

```
Red Hat: drwxrwxr-x  2  root  mail  1024 Dec  5 11:16  /var/spool/mail
Fedora:  drwxrwxr-x  2  root  mail  4096 Mar 17 08:42  /var/spool/mail
SUSE:    drwxrwxrwt  2  root  root  4096 Aug  2 23:25  /var/spool/mail
Debian:  drwxrwsr-x  2  root  mail  4096 Aug  3 16:17  /var/mail
Ubuntu:  drwxrwsr-x  2  root  mail  4096 Jan  8 03:22  /var/mail
```

Права доступа к каталогу хранилища в SUSE могут показаться слишком великодушными, однако сами файлы в хранилище имеют режим доступа 660 и принадлежат группе root. Каталоги, у которых установлен sticky-бит (обозначен буквой t в строке режима), не позволяют пользователям удалять чужие файлы, даже если пользователь имеет право записи в каталог. Правда, злоумышленник может переполнить каталог "мусором" или создать в нем почтовый ящик другого пользователя.

📖 О sticky-бите рассказывалось в разделе 5.5.

Агенты доступа

Такие программы, как imapd и spop, являются агентами доступа для пользователей Windows, Macintosh или Linux, чья почта доставляется на Linux-сервер, а затем загружается с него по протоколу IMAP (Internet Message Access Protocol — протокол доступа к сообщениям в сети Internet) или POP (Post Office Protocol — почтовый протокол), соответственно. Оба протокола описываются в конце раздела 18.3.

Агенты подачи почты

Необходимость в программах этого класса возникла в связи с ростом объема почтовых хостов. На большом хосте транспортный агент вынужден тратить много времени на предварительную обработку сообщений. В частности, он должен:

- проверять, являются ли имена хостов полностью определенными;
- модифицировать заголовки сообщений, полученных от неправильно работающих пользовательских агентов;
- вести журнал ошибок;
- переписывать заголовки и т.д.

В документе RFC2476 была сформулирована идея отделить от транспортного агента особый агент подачи почты с целью распределения нагрузки и повышения общей производительности. Суть идеи в том, что агент подачи работает на другом порту и выступает в качестве регистратора новых сообщений, поступающих в почтовую систему от локальных пользовательских агентов. Он делает всю предварительную работу, включая проверку ошибок, прежде чем сообщение будет отправлено транспортным агентом.

[5] Системы, которые осуществляют доставку почты путем изменения принадлежности файлов с помощью команды chown, причем не в режиме суперпользователя, должны иметь групповое право записи в этот каталог.

В частности, агент подачи должен убедиться в том, что все имена хостов являются полностью определенными. Кроме того, перед добавлением локального доменного имени к адресу он проверяет, являются ли имена локальных хостов легитимными. Агент подачи также исправляет заголовки сообщений, если в них есть ошибки (и добавляет заголовки, если они отсутствуют). Чаще всего добавляются заголовки "From" и "Date" и корректируется заголовок "Message-Id". Наконец, агент подачи может изменить адрес отправителя, подставив вместо регистрационного имени пользователя заданное внешнее имя (например, в формате *Имя_Фамилия*).

Чтобы этот механизм работал, пользовательские агенты должны быть сконфигурированы на подключение к агенту подачи через порт 587, а не через порт 25, который традиционно применяется для электронной почты. Если пользовательский агент по какой-то причине не может работать с портом 587, существует другой путь: вызывать агент подачи через порт 25, но не на том сервере, на котором работает транспортный агент. Необходимо также сконфигурировать транспортный агент таким образом, чтобы он не проделывал повторно работу, выполненную агентом подачи. Дублирование не скажется на правильности обработки почты, но отразится на производительности.

По умолчанию программа `sendmail` может действовать и как агент подачи, и как транспортный агент. Начиная с версии 8.10 один экземпляр программы способен прослушивать оба порта: 25 и 587. Пользовательские агенты часто вызывают программу `sendmail` с флагами, которые указывают на необходимость приема сообщений (`-bs` или `-bm`), либо вообще без флагов. В последнем случае поведение программы соответствует режиму `-bm`. Программа проверяет способ своего вызова и становится агентом подачи почты (в случае флагов `-bs` и `-bm`) либо транспортным агентом (в случае флага `-bd`).

Если пользовательский агент непосредственно устанавливает SMTP-соединение, то нужно настроить его на использование порта 587, чтобы включить в работу агент подачи почты.

18.2. Структура почтового сообщения

Прежде чем перейти к вопросам конфигурирования программы `sendmail`, рассмотрим структуру электронного сообщения. Оно состоит из трех частей:

- конверт;
- заголовки;
- тело.

Адреса конверта определяют, куда должно быть доставлено сообщение или куда его требуется возвратить в случае, если доставка невозможна. Обычно они соответствуют адресам, указанным в заголовках "From" и "To", но это правило нарушается в случае списка рассылки. Адреса конверта обрабатываются агентом подачи почты отдельно. Конверт невидим для пользователя и является внутренним средством программы `sendmail`.

Заголовки — это набор пар свойство/значение, отформатированных в соответствии с документом RFC2822. В них содержится самая разнообразная информация о сообщении, включая дату и время его отправки, а также сведения о транспортных агентах, через которые оно прошло на своем пути. Заголовки являются неотъемлемой частью сообщения, но пользовательские агенты часто скрывают некоторые менее интересные заголовки при отображении сообщения.

Тело сообщения — это та информация, которую, собственно, и требуется переслать. Оно должно содержать ASCII-текст, однако часто он представляет собой двоичные данные в специальной почтовой кодировке.

В разделах, посвященных вопросам конфигурирования, мы иногда будем говорить об отправителе и получателях конверта, а иногда — об отправителе и получателях, указанных в заголовках. Там, где из контекста не ясно, о чем идет речь, тип адреса будет конкретизирован.

Адресация почты

Механизм локальной адресации прост, потому что регистрационное имя пользователя является уникальным идентификатором. Почтовый Internet-адрес столь же прост: *пользователь*@*хост.домен* или *пользователь*@*домен*. Когда-то давно применялись адреса и других типов (табл. 18.3).

Таблица 18.3. Примеры устаревших адресов

Тип адреса	Пример	Современная форма
UUCP	mcvax!uunet!ucbvax!hao!boulder!lair!evi	evi@lair
Маршрутно-зависимый	<@*хост1*,@*хост2*,... @*хостN*:*пользователь*@*конечный_хост*>	*пользователь*@*конечный_хост*
Процентный	*пользователь*%*хост1*%*хост2*@*хост3*	*пользователь*@*хост1*

Основные сложности в конфигурации программы sendmail связаны с существовавшей ранее необходимостью управлять указанными видами адресов. Каждая из этих форм адресации связана с ретрансляцией почтовых сообщений, от которой постепенно отказываются из-за активности спамеров. “Процентный” адрес (указан последним в таблице) является самым любимым средством спамеров, которые хотят скрыть свою личность или переслать почту через ничего не подозревающие компьютеры-посредники. Подробнее о работе с такими адресами можно узнать в документации к программе sendmail.

Интерпретация заголовков почтовых сообщений

Каждое электронное сообщение начинается с нескольких строк, которые называются заголовками и содержат информацию о сообщении. Любой заголовок обозначен ключевым словом, таким как “To”, “From” или “Subject”, после которого следуют символ двоеточия и содержимое заголовка. Формат стандартных заголовков определен в документе RFC2822. Однако допускаются и заголовки других типов. Любой заголовок с префиксом “X-” почтовой системой игнорируется, но пересылается вместе с сообщением. Следовательно, можно добавить в сообщение заголовок наподобие “X-Joke-of-the-Day” и не беспокоиться о том, способна ли почтовая система обработать его[6].

Одни заголовки добавляются пользовательским агентом, а другие — транспортным. Некоторые заголовки отражают путь сообщения в почтовой системе. Многие пользовательские агенты скрывают эти “неинтересные” заголовки от пользователей, но, как правило, всегда имеется возможность заставить агента показать их. Чтение заголовков

[6] С технической точки зрения можно добавить совершенно произвольный заголовок, поскольку в процессе маршрутизации почты используются адреса конверта, а заголовки игнорируются.

становится важным делом, когда вас заваливают спамом и нужно проследить, откуда он поступает. Ниже приведен блок заголовков простого сообщения.

```
From trent Fri, 30 Jun 2006 20:44:49 -0600
Received: from bull.atrust.com (bull.atrust.com [127.0.0.1]) by bull.atrust.com
    (8.13.1/8.13.1) with ESMTP id k612inkG001576 for <ned@bull.atrust.com>;
    Fri, 30 Jun 2006 20:44:49 -0600
Date: Fri, 30 Jun 2006 20:44:48 -0600
From: trent@atrust.com
Message-Id: <200607010244.k612im9h001575@bull.atrust.com>
To: ned@bull.atrust.com
Cc: steve@bull.atrust.com
Subject: Yonder Mountain

--------- тело сообщения ---------
```

Это сообщение не покидало локальный компьютер. Его отправителем является пользователь `trent`, а получателем — пользователь `ned`. Первая строка "From" добавлена программой `mail.local`, которая была агентом доставки в данном примере. Заголовки "Subject" и "Cc" добавлены программой чтения почты пользователя `trent`, как и заголовки "To", "From" и "Date", очевидно. Программа `sendmail` (транспортный агент) создает заголовки "To", "From" и "Date" только в том случае, если они не были созданы пользовательским агентом. Каждый компьютер (точнее, каждый транспортный агент), участвующий в процессе передачи сообщения, добавляет свой заголовок "Received".

Заголовки электронного сообщения содержат массу информации о том, где было сообщение, насколько долго оно там оставалось и когда, наконец, было доставлено адресату. Ниже более подробно описаны заголовки почтового сообщения, посланного через Internet. Мы расскажем о назначении различных заголовков и о том, какие программы их добавляют. Номера строк (слева) даны для удобства и не являются частью сообщения. Некоторые заголовки разбиты на несколько строк, чтобы пример поместился на странице.

```
1: From eric@knecht.sendmail.org
```

Первую строку добавила программа `/bin/mail` или `mail.local` в процессе конечной доставки, чтобы отделить это сообщение от других сообщений, находящихся в почтовом ящике пользователя-получателя. Некоторые программы чтения почты распознают границы сообщений путем поиска пустой строки, за которой следуют символы "From " (здесь обязательно наличие конечного пробела). Эта строка не существует до тех пор, пока сообщение не будет доставлено, и ее не следует путать с заголовком "From:". Многие программы чтения почты не отображают строку "From ", поэтому ее можно и не увидеть.

```
2: Return-Path: eric@knecht.Neophilic.COM
```

В строке 2 указан адрес для возврата сообщения. Он может отличаться от адреса, указанного ниже в заголовке "From:". Сообщения об ошибке посылаются по адресу, заданному в строке "Return-Path", которая содержит адрес конверта отправителя.

```
3: Delivery-Date: Mon, 06 Aug 2001 14:31:07 -0600
```

В строке 3 показано время доставки сообщения в локальный почтовый ящик пользователя `evi`. Учтено смещение (-6 часов) от нулевого меридиана для данного часового пояса.

```
4: Received: from anchor.cs.colorado.edu (root@anchor.cs.colorado.edu
      [128.138.242.1]) by rupertsberg.cs.colorado.edu (8.10.1/8.10.1)
      with ESMTP id f76KV7J25997 for <evi@rupertsberg.cs.colorado.edu>;
      Mon, 6 Aug 2001 14:31:07 -0600 (MDT)
5: Received: from mroe.cs.colorado.edu
      (IDENT:root@mroe.cs.colorado.edu [128.138.243.151]) by
      anchor.cs.colorado.edu (8.10.1/8.10.1) with ESMTP id f76KV6418006
      for <evi@anchor.cs.colorado.edu>; Mon, 6 Aug 2001 14:31:06
      -0600 (MDT)
6: Received: from knecht.Neophilic.COM (knecht.sendmail.org
      [209.31.233.176]) by mroe.cs.colorado.edu (8.10.1/8.10.1)
      with ESMTP id f76KV5Q17625 for <evi@anchor.cs.colorado.edu>;
      Mon, 6 Aug 2001 14:31:05 -0600 (MDT)
7: Received: from knecht.Neophilic.COM (localhost.Neophilic.COM
      [127.0.0.1]) by knecht.Neophilic.COM
      (8.12.0.Beta16/8.12.0.Beta17) with ESMTP id f76KUufp084340 for
      <evi@anchor.cs.colorado.edu>; Mon, 6 Aug 2001 13:30: 56
      -0700 (PDT)
```

В строках 4–7 содержится информация о продвижении сообщения по различным системам на пути к почтовому ящику пользователя. Каждый компьютер, который обрабатывает электронное сообщение, добавляет к его заголовку строку "Received". Новые строки вставляются сверху, поэтому, читая строки "Received" последовательно, можно проследить путь сообщения в обратном порядке (от получателя к отправителю). Однако если сообщение является частью спама, то единственная строка "Received", которой можно доверять, — это строка, сгенерированная локальным компьютером.

Каждая строка "Received" содержит имя хоста-отправителя, имя хоста-получателя, данные о версии программы `sendmail` (или другого транспортного агента) на хосте-получателе, уникальный идентификатор сообщения на хосте-получателе, данные о пользователе-получателе (если он один), дату, время и, наконец, разность между временем в данном часовом поясе и временем по Гринвичу. Эти сведения берутся из внутренних макроконстант программы `sendmail`. Рассмотрим приведенные выше строки, начиная с последней, и проследим путь сообщения от отправителя к получателю.

Как показано в строке 7, сообщение первоначально поступило из интерфейса `localhost` (который был выбран пользовательским агентом Эрика Оллмана для установления соединения) хоста `knecht` на внешний интерфейс этого же хоста через псевдоустройство обратной связи. В строке 6 отражен тот факт, что компьютер `knecht` направил сообщение по адресу `mroe.cs.colorado.edu`, несмотря на то что заданный пользователем адрес — `evi@anchor.cs.colorado.edu` (см. заголовок строки 9). Быстрая проверка посредством утилиты `dig` или `nslookup` показывает, что хост `anchor` имеет запись MX, которая ссылается на компьютер `mroe`. Это и стало причиной перенаправления сообщения. На компьютере `knecht` работала программа `sendmail` версии 8.12.0Beta16.

📖 Подробнее о записях MX рассказывалось в разделе 15.7.

На компьютере `mroe` работала программа `sendmail` версии 8.10.1; она присвоила сообщению идентификатор очереди f76KV5Q17625, который действителен, пока сообщение находится в очереди на данном хосте. Далее компьютер `mroe` переслал сообщение по адресу `anchor.cs.colorado.edu` (строка 5), что выглядит весьма странным, ведь перед этим сообщение попало на хост `mroe` как раз вместо хоста `anchor` из-за наличия записи MX. Причиной этого кажущегося парадокса является использование в доме-

не cs.colorado.edu конфигурации, называемой “расщепление DNS”. Запись MX на хосте anchor, которую видят пользователи за пределами домена, направляет входящую почту на главный компьютер (mroe). В то же время внутри домена cs.colorado.edu видна другая запись. Внутренняя версия записи указывает сначала на сам компьютер anchor, а затем — на компьютер mroe в качестве резервного сортировщика почты.

Поступив на хост anchor, сообщение сразу же перенаправляется, на этот раз на компьютер rupertsberg (строка 4). Очередная смена адреса вызвана наличием псевдонима (более подробно почтовые псевдонимы рассматриваются в разделе 18.4).

Псевдонимы играют важную роль в пересылке почты. Псевдоним ставит имя пользователя в соответствие другому объекту, например имени этого же пользователя на другом компьютере, группе пользователей или даже другому варианту написания его имени. Таким образом, на основании содержимого одних лишь заголовков невозможно определить, почему сообщение отклонилось от своего пути. Как и в случае записей MX, нужно искать внешние источники информации.

Строки 5 и 6 содержат текст “for <evi@anchor.cs.colorado.edu>”. Он показывает, как выглядит адрес получателя, когда сообщение поступает на локальный компьютер. Эта информация полезна, если нужно исключить пользователя из списка рассылки, отправив сообщение об отказе с того компьютера, с которого была сделана подписка (возможно, много лет тому назад), либо узнав адрес этого компьютера и включив его в тело сообщения.

В конечном заголовке “Received” (строка 4) содержится текст “for <evi@rupertsberg.cs.colorado.edu>”. Значение макроконстанты программы sendmail, хранящей адрес получателя, было изменено из-за наличия псевдонима на хосте anchor. Локальный агент доставки на компьютере rupertsberg поместил сообщение в почтовый ящик пользователя evi.

```
8: Message-Id: <200108062030.f76KUufp084340@knecht.Neophilic.COM>
```

В строке 8 содержится идентификатор сообщения, который отличается от идентификатора очереди и уникален во всемирной системе электронной почты. Он добавляется к сообщению, когда оно попадает в почтовую систему.

```
 9: To: evi@anchor.cs.colorado.edu
10: From: Eric Allman <eric@Sendmail.ORG>
11: X-URL: http://WWW.Sendmail.ORG/~eric
12: Subject: example message for Evi
13: Date: Mon, 06 Aug 2001 13:30:56 -700
14: Sender: eric@knecht.Neophilic.COM
```

Строки 9, 10, 12, 13 и 14 — стандартные. Заголовок “Subject” необязателен, но большинство пользовательских агентов включает его. Строка “To” содержит адрес основного получателя (или получателей). В заголовке “From” отправитель обозначен как eric@sendmail.org, хотя из заголовков “Received” следует, что отправитель находится в домене neophilic.com. Все дело в том, что с компьютером knecht Эрика Оллмана связано несколько виртуальных доменов, не только sendmail.org.

В строке “Date” приведены дата и время отправки сообщения. В данном случае время отправки почти совпадает с указанным в заголовках “Received”, несмотря на то что значения времени фиксировались разными таймерами.

В строке 11 указан URL-адрес домашней Web-страницы Эрика. Обратите внимание на префикс “X-”: это нестандартный заголовок. Когда только появилась электронная почта, не было ни Web-страниц, ни URL-адресов.

Строки “Received” обычно добавляются транспортными агентами (если только это не подделка), а другие заголовки создаются пользовательскими агентами. Некоторые пользовательские агенты работают некорректно и не добавляют нужных заголовков. В этом случае на помощь приходит программа `sendmail`.

Первая строка “Received” обычно создается компьютером-отправителем, когда сообщение поступает на выходной интерфейс. Иногда эта строка включает фразу “ident”, которая содержит регистрационное имя отправителя. Это имя должно совпадать с указанным в строке “From” (если последнюю не подменили). В нашем примере на компьютере Эрика `knecht` не запущен демон, реализующий данную функцию (`identd`), поэтому фраза “ident” отсутствует.

На рис. 18.2 изображен путь сообщения в системе электронной почты. Показано, какие действия, где и какими программами выполнялись.

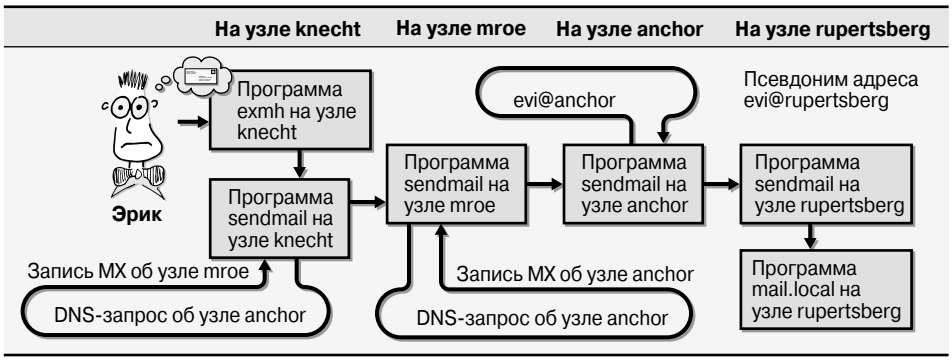


Рис. 18.2. Путь сообщения от Эрика

Нетрудно заметить, что программа `sendmail` в данной схеме играет роль рабочей лошадки. Эта программа занималась сообщением с момента его выхода из программы `exmh` вплоть до прибытия на компьютер `rupertsberg` для доставки адресату.

18.3. Принципы организации электронной почты

Соблюдение принципов работы с электронной почтой, описанных в этой главе, является обязательным условием нормального администрирования почтовых систем в средних и крупных организациях. Естественно, ими можно руководствоваться и в небольших организациях. Есть несколько основных факторов, которые позволяют значительно облегчить решение административных задач. Это:

- наличие серверов для входящей и исходящей почты или, в случае очень крупных организаций, иерархии серверов;
- наличие почтового каталога для каждого пользователя в организации;
- поддержка протоколов IMAP или POP[7] для интеграции персональных компьютеров, систем Macintosh и удаленных клиентов.

[7] В настоящее время предпочтение отдается протоколу IMAP. Если вы поддерживаете доступ к почте из-за пределов вашей организации, нужно использовать версии этих протоколов с шифрованием с помощью SSL (IMAPS или POPS, соответственно). Более подробно об этом см. на с. 542.

Каждый из этих факторов будет подробно проанализирован ниже. Кроме того, необходимо добиться хорошего взаимодействия и функционирования других подсистем и средств: записи MX базы данных DNS должны быть правильными, брандмауэры должны пропускать входящую и исходящую почту, должны быть определены компьютеры, на которых организованы хранилища сообщений, и т.д.

📖 Подробнее о записях MX рассказывалось в разделе 15.7.

Почтовые серверы решают четыре задачи:

- принимают исходящую почту от пользовательских агентов и направляют ее в почтовую систему;
- принимают входящую почту из внешнего мира;
- доставляют сообщения в почтовые ящики пользователей;
- позволяют пользователям получать доступ к почтовым ящикам посредством протокола IMAP или POP.

В небольшой организации серверы, реализующие указанные функции, могут работать на одном компьютере. В крупных сетях это должны быть отдельные компьютеры. Настройка брандмауэра значительно упростится, если входящая почта будет поступать на один компьютер, и вся исходящая почта тоже будет проходить через один компьютер.

В некоторых организациях внешняя почта поступает на прокси-сервер. В действительности прокси-сервер не обрабатывает почту, а лишь принимает и накапливает ее у себя. Отдельный процесс затем пересылает эту почту программе `sendmail` для последующей транспортировки и обработки. Примерами прокси-серверов для программы `sendmail` являются демоны `smtpd` и `smtpfwdd` (доступны на Web-сайте `www.obtuse.com`). Оба они свободно распространяются с открытым исходным кодом. Демон `smtpd` способен фильтровать входящую почту на основании списков доступа. Ни в одном из наших демонстрационных дистрибутивов эти демоны не включены в стандартный комплект поставки.

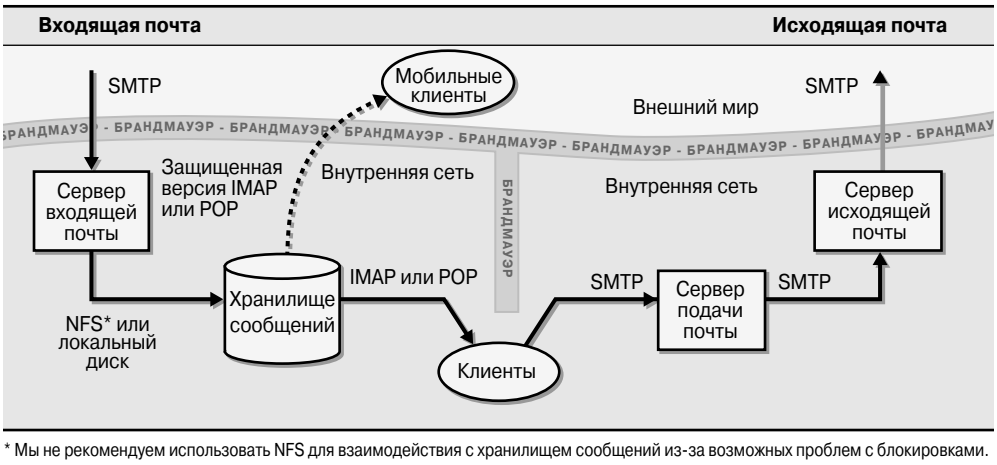
Почтовые серверы

Для почтового сервера необходимо подобрать надежный и стабильно работающий компьютер. Ниже мы опишем структуру почтовой системы, которая, как нам кажется, является легко масштабируемой, хорошо управляемой и безопасной. Задачи управления входящей и исходящей почтой в ней решают специально выделенные для этих целей серверы. Один из вариантов такой структуры приведен на рис. 18.3.

Почтовая система, изображенная на рис. 18.3, имеет одну точку получения данных из внешнего мира — это почтовый сервер, подключенный к сети Internet. Сервер исходящей почты тоже непосредственно подключен к Internet, но менее уязвим, поскольку сам устанавливает соединения, а не принимает запросы от внешних хостов. Следует тщательно контролировать сервер входящей почты, регулярно устанавливать на нем "заплаты" и последние версии программы `sendmail`, включающие фильтры для борьбы со спамом.

Сервер исходящей почты должен не менее хорошо контролироваться. Он может содержать собственные фильтры от спама, чтобы выявлять локальных пользователей, являющихся инициаторами спама. Кроме того, когда есть риск утечки ценной информации из внутренней сети, использование единственного сервера, через который проходит исходящая почта, упрощает реализацию правил безопасности и контроль над их

соблюдением. Если в организации используются большие списки рассылки, на сервере исходящей почты можно установить средства программы `sendmail`, предназначенные для повышения производительности (об этом рассказывается в разделе 18.13).



* Мы не рекомендуем использовать NFS для взаимодействия с хранилищем сообщений из-за возможных проблем с блокировками.

Рис. 18.3. Структура почтовой системы (вариант 1)

Серверы входящей и исходящей почты можно реплицировать, если того требуют объемы проходящей почты. К примеру, группой серверов входящей почты может управлять один компьютер, осуществляющий распределение нагрузки; другой вариант — использовать записи MX, которые будут делать приблизительно то же самое. Если есть несколько серверов исходящей почты, то разные клиенты смогут пересылать почту через разные серверы. Но учтите, что данные между серверами входящей и исходящей почты не должны передаваться напрямую. Серверы следует отделить друг от друга посредством внутреннего брандмауэра.

В особенно крупных организациях репликации серверов, очевидно, не избежать. Для поиска почтовых ящиков пользователей и направления почты в соответствующие хранилища придется ввести дополнительный уровень маршрутизации (возможно, с помощью протокола LDAP). На этом уровне может также выполняться фильтрация спама и вирусов, прежде чем сообщения будут попадать в почтовые ящики.

Провайдеры, проектирующие почтовые системы для обслуживания клиентов, должны установить дополнительный сервер, на который будут указывать резервные записи MX клиентов, и который будет управлять списками рассылки. Этот компьютер должен принимать почту и распространять ее среди клиентов. Его следует жестко контролировать, чтобы почта направлялась только аутентифицированным клиентам, и отделить брандмауэром от серверов входящей и исходящей почты.

В пользовательских Linux-системах достаточно установить программу `sendmail` в минимальной конфигурации, рассчитанной на пересылку исходящей почты соответствующему серверу для обработки. Принимать почту из Internet нет необходимости. Однако иногда от этой жесткой модели отступают и разрешают любым компьютерам посылать почту непосредственно в Internet. В любом случае на всех клиентских компьютерах программа `sendmail` может быть сконфигурирована одинаково. Организовать распространение такой конфигурации можно с помощью утилиты `rdist` или `rsync`.

Подробнее вопросы рассылки файлов рассматривались в разделе 17.3.

В организациях, где используются программы наподобие Microsoft Exchange и Lotus Notes и где нежелательно предоставлять этим программам прямой выход в Internet, можно придерживаться структуры почтовой системы, изображенной на рис. 18.4.

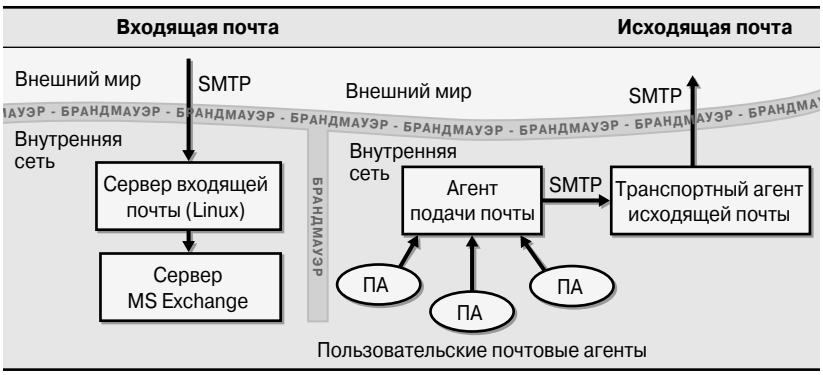


Рис. 18.4. Структура почтовой системы (вариант 2)

Независимо от того, какая структура почтовой системы выбрана, позаботьтесь о том, чтобы конфигурация программы `sendmail`, записи MX и правила работы брандмауэра были согласованы друг с другом.

Почтовые каталоги пользователей

Пользователю, несомненно, удобнее принимать и хранить всю свою почту на одном компьютере, даже если ему нужно получать доступ к этой почте из нескольких систем. Эта возможность реализуется с помощью файла `aliases` или базы данных LDAP (описана в конце раздела 18.5). Удаленный доступ к почтовым ящикам каждого пользователя можно организовать посредством протокола IMAP или POP.

Схема создания псевдонимов, которой мы пользуемся, позволяет иметь один и тот же файл псевдонимов на всех компьютерах домена. Для администратора это большой плюс. (Мы предполагаем, что регистрационные имена и идентификаторы пользователей уникальны на всех компьютерах, и настоятельно рекомендуем проводить такую политику.)

В некоторых организациях систему электронной почты централизуют, экспортируя каталог `/var/spool/mail` через NFS. Однако проблемы блокировки файлов в NFS могут привести к потере или повреждению файлов, хранящих почту пользователей. Можно, конечно, потом обвинять NFS, программу `sendmail` или агенты доставки, но это вряд ли поможет несчастному пользователю, чей почтовый ящик поврежден (программа `sendmail`, кстати, всегда невиновна, потому что она не осуществляет фактическую доставку почты).

Некоторые варианты реализации NFS (например, с использованием специализированных файловых серверов) включают в себя программу управления блокировками, которая работает нормально. Но в большинстве реализаций блокировки либо не поддерживаются, либо поддерживаются некорректно. В одних организациях проблему блокировок просто игнорируют, надеясь на авось, а в других — требуют, чтобы пользователи читали почту на файловом сервере.

Наш совет: не стоит организовывать совместный доступ к каталогу `/var/spool/mail` через NFS.

Протоколы IMAP и POP

IMAP и POP — это протоколы загрузки почты на пользовательские компьютеры, когда те подключаются к сети. Данные протоколы являются идеальным средством управления почтой, особенно для компьютеров, которые не всегда подключены к сети либо из-за того, что их выключают после работы, либо из-за того, что они находятся дома и соединяются с сетью по телефонной линии. В обоих протоколах пароль обязателен; убедитесь, что он не передается по сети в незашифрованном виде (используйте протоколы IMAPS и POPS).

Наши симпатии на стороне IMAP (Internet Message Access Protocol — протокол доступа к сообщениям в сети Internet). Этот протокол обеспечивает доставку сообщений по одному, а не всех сразу. Такой метод лучше для тех пользователей, которые подключены к низкоскоростной сети или имеют портативные компьютеры. Протокол IMAP особенно удобен при отборе писем с гигантскими вложениями: можно просматривать заголовки сообщений и не загружать письма с вложениями до тех пор, пока не будет выяснено их содержимое.

Протокол управляет почтовыми папками в нескольких системах, например на почтовом сервере и персональном компьютере. Почта, остающаяся на Linux-сервере, может быть объектом резервного копирования. Множество полезной информации о протоколе IMAP и его реализациях находится на Web-сайте `www.imap.org`.

POP (Post Office Protocol — почтовый протокол) реализует похожую модель, в которой, однако, с сервера на персональный компьютер загружаются все сообщения. При этом они могут удаляться с сервера (зачастую безвозвратно) или сохраняться на нем (в этом случае хранилище почты будет постоянно разрастаться). Метод доставки всей почты за раз неоптимален для сети и не очень гибок с точки зрения пользователя. Это особенно проявляется в коммутируемых соединениях.

Оба протокола с трудом справляются с ситуацией, когда пользователь, ни разу не удалявший полученные сообщения, пытается все их загрузить. В случае протокола IMAP потребуется немало времени, чтобы просмотреть заголовки всех сообщений, а в случае протокола POP по сети будет передано все содержимое почтового ящика. Постарайтесь объяснить пользователям, как важно удалять сообщения с сервера и хранить их в локальных папках.

Приемлемая реализация POP3, современной версии протокола, доступна по адресу `www.eudora.com/products/unsupported/qpopper`. Сервер `qpopper` реализует механизмы аутентификации TLS/SSL между сервером и клиентом и шифрует сообщения.

В сети Internet можно найти множество других серверов POP3 для Linux; выбирайте только тот сервер, который имеет активную поддержку.

Программное обеспечение сервера IMAP доступно по адресу `www.washington.edu/imap`. Конфигурировать сервер не требуется, нужно лишь поместить соответствующие записи в файлы `/etc/services` и `/etc/inetd.conf`. Кроме того, следует убедиться в том, что брандмауэр не препятствует работе сервера. С протоколом IMAP раньше было связано много проблем безопасности, поэтому следуйте советам организации CERT и старайтесь пользоваться самыми последними версиями программного обеспечения, особенно если в бюллетенях, посвященных вопросам безопасности, говорится о том, что в вашем дистрибутиве Linux есть бреши.

Специалисты университета Карнеги-Меллона разработали IMAP-сервер Cyrus, который поддерживает также протокол POP. Этот сервер нравится нам больше, чем реализация университета штата Вашингтон (UW IMAP), так как он предлагает конечным пользователям большие функциональные возможности и обладает высокой производительностью.

Самым свежим пакетом, реализующим протоколы IMAP и POP, является Dovecot. Он был написан с учетом всех строгих требований, предъявляемых к написанию кода, которые позволяют повысить его безопасность, по крайней мере, теоретически. Этот пакет имеет интересные функциональные особенности. Например, он может хранить электронную почту в базе данных SQL, а не в файловой системе. Пока что этот пакет не может похвастаться серьезными успехами или инсталлированной базой Cyrus; тем не менее, он стоит того, чтобы вы изучили и оценили все его достоинства.

Во всех наших демонстрационных дистрибутивах имеется IMAP-сервер `imapd`, а также клиент `fetchmail`, понимающий протоколы IMAP и POP. В Red Hat демон `imapd` представляет собой реализацию Cyrus IMAP, а в SUSE, Debian и Ubuntu — UW IMAP. В Red Hat есть также POP-сервер `pop3d`. Чтобы не отставать от конкурентов, разработчики SUSE включили в дистрибутив три POP-сервера: `qpopper` (переименован в `popper`), `pop2d` и `pop3d`. В Debian существует ряд средств для управления почтовыми ящиками по протоколу IMAP; введите команду `man -k imap`, чтобы получить их список.

18.4. Почтовые псевдонимы

Псевдонимы позволяют системному администратору или отдельным пользователям переадресовывать почту[8]. Их можно применять для задания списков рассылки, для пересылки почты между компьютерами, а также для того, чтобы к пользователям можно было обращаться по нескольким именам. Псевдонимы обрабатываются рекурсивно, поэтому могут указывать на объекты, которые, в свою очередь, тоже являются псевдонимами.

Программа `sendmail` поддерживает несколько механизмов создания псевдонимов, а именно:

- разнообразные карты плоских файлов, которые без труда могут устанавливать пользователи и системные администраторы (например, `/etc/aliases`);
- продвигаемые поставщиком устаревшие системы распространения файлов, такие как NIS и NIS+ от Sun[9] и NetInfo от Apple;
- различные базы данных почтовой маршрутизации;
- протокол LDAP.

До настоящего времени, в малых и средних организациях проще всего устанавливать плоские файлы (например, `/etc/aliases`; о них мы поговорим далее в этой главе). Если в крупной организации нужно придерживаться концепции личных почтовых каталогов пользователей, то рекомендуем реализовать ее путем хранения псевдонимов на сервере LDAP.

[8] По техническим причинам псевдонимы настраивает только системный администратор. Пользователи управляют маршрутизацией почты при помощи файла `.forward`, который в действительности не имеет прямого отношения к псевдонимам, но мы объединили здесь оба этих средства.

[9] Поддержка системы NIS+ была официально прекращена компанией Sun в 2005 году. Вместо этой системы теперь поддерживается LDAP.

Протокол LDAP подробно описан в трех местах этой главы, а также в главе 17. В конце этого раздела мы расскажем о том, как работает протокол, затем (в разделе 18.5) рассмотрим вопросы конфигурирования протокола LDAP в `sendmail`. Однако прежде всего следует описать традиционные способы назначения псевдонимов.

Псевдонимы могут быть определены в трех местах (к сожалению, с использованием трех разных синтаксисов):

- в файле конфигурации пользовательского агента (пользователем-отправителем);
- в общесистемном файле псевдонимов `/etc/mail/aliases` (системным администратором);
- в пользовательском файле переадресации `~/.forward` (пользователем-получателем)[10].

Пользовательский агент ищет псевдонимы в конфигурационных файлах пользователя и раскрывает их до подачи сообщения в почтовую систему. Транспортный агент, т.е. программа `sendmail`, ищет псевдонимы в глобальном файле `aliases`, а затем в файлах `.forward` получателей. Псевдонимы применяются только к тем сообщениям, которые программа `sendmail` считает локальными[11].

Вот несколько примеров определения псевдонимов в файле `aliases`:

```
nemeth: evi
evi: evi@mailhub
authors: evi,garth,trent
```

Первая строка говорит о том, что почту, поступающую на имя `nemeth`, следует доставлять пользователю `evi` на локальном компьютере. Вторая — что всю почту, поступающую на имя `evi`, следует доставлять на компьютер `mailhub`. И, наконец, третья строка определяет, что почту, адресованную группе `authors`, следует доставлять пользователям `evi`, `garth` и `trent`. Благодаря рекурсии почта, посланная на имя `nemeth`, попадет по адресу `evi@mailhub`.

Путь к файлу `aliases` задается в конфигурации программы `sendmail`. В Red Hat, SUSE и Ubuntu он называется `/etc/aliases`, хотя его "стандартное" местонахождение — `/etc/mail/aliases`. Допускается наличие нескольких файлов `aliases`, а также использование альтернативных механизмов хранения псевдонимов, например службы NIS или специальных баз данных.

📖 Подробнее о службе NIS рассказывалось в главе 17.

Записи файла `aliases` имеют следующий формат:

`локальное_имя``: `*`получатель1`*`,`*`получатель2`*`,...`

где *`локальное_имя`* — это адрес, который сравнивается с адресом входящего сообщения, а список получателей содержит либо адреса получателей, либо другие псевдонимы. Строки с отступом считаются продолжением предыдущих строк.

Для системы электронной почты файл `aliases` имеет приоритет над файлом `/etc/passwd`, поэтому существование записи

```
david: david@somewhere_else.edu
```

[10] Файл `~/.forward` просматривается программой `sendmail` по умолчанию. Можно задать другой путь к этому файлу, воспользовавшись переменной `ForwardPath`.

[11] Вообще-то, это не совсем так. Если в конфигурацию почтового агента, работающего по протоколу SMTP, добавить установку `F=A`, то можно реализовать псевдонимы для удаленных адресов.

не позволит локальному пользователю david вообще когда-либо получить почту. По этой причине при добавлении новых пользователей (вручную или с помощью сценария adduser) необходимо проверять как файл passwd, так и файл aliases.

Файл /etc/mail/aliases всегда должен содержать псевдоним postmaster, который обеспечивает пересылку сообщений администратору почтовой системы. Точно так же, псевдонимы abuse используются в том случае, если какой-то человек, который не работает в вашей организации, хочет пообщаться с вами относительно спама или подозрительного поведения в сети, исходящего из вашей организации. Кроме того, должен существовать псевдоним для автоматических сообщений программы sendmail. Обычно он называется Mailer-Daemon и эквивалентен псевдониму postmaster.

Почту, приходящую на имя root, необходимо переадресовывать администратору или другому человеку, осуществляющему каждодневный контроль системы. Учетным записям bin, sys, daemon, nobody и hostmaster (и всем остальным имеющимся в системе псевдопользовательским учетным записям) тоже должны соответствовать псевдонимы, обеспечивающие пересылку почты реальным пользователям. Файл sendmail/aliases из дистрибутива программы является хорошим шаблоном для создания псевдонимов системного уровня. В нем можно найти советы по поводу безопасности и примеры обработки пользовательских запросов.

Программа sendmail выявляет "петли" псевдонимов, способные привести к бесконечной циклической пересылке почты. Для этого программа подсчитывает число строк "Received" в заголовке сообщения и, как только это число превышает установленный лимит (обычно он равен 25)[12], возвращает сообщение отправителю. Каждое посещение сообщением нового компьютера на жаргоне разработчиков sendmail называется "переходом" (hop), а возвращение сообщения отправителю — "рикошетом" (bounce)[13].

Помимо списков пользователей псевдонимы могут ссылаться на:

- файл, содержащий список адресов;
- файл, в который должны добавляться сообщения;
- программу, на вход которой должны передаваться сообщения.

Поскольку отправитель сообщения полностью определяет его содержимое, указанные выше объекты доставки часто используются хакерами. В результате программа sendmail стала очень подозрительна в отношении владельцев и прав доступа к таким файлам и командам. Чтобы избавить программу от этой подозрительности, нужно поменять значение опции DontBlameSendmail ("не обвиняй sendmail" — это название следует понимать так: не обвиняй программу, если изменение опции приведет к возникновению проблем). К сожалению, сообщения об ошибках, которые генерируются программой при обнаружении нарушений, связанных с правами доступа и владельцами файлов, не всегда понятны.

[12] Значение 25 задано по умолчанию. Этот параметр можно изменить в файле конфигурации.

[13] В этой главе мы несколько непоследовательны в терминологии, иногда называя возвращаемое сообщение "рикошетом", иногда — "сообщением об ошибке". На самом деле имеется в виду уведомление о состоянии доставки (Delivery Status Notification, DSN). Обычно оно означает, что сообщение не удалось доставить и поэтому оно возвращается отправителю.

Загрузка списков рассылки из файла

Директива `:include:` является отличным средством управления пользовательскими списками рассылки. Она позволяет не перечислять компоненты псевдонима непосредственно в файле `aliases`, а брать их из внешнего файла. Этот файл можно менять без вмешательства системного администратора, отвечающего за глобальный файл `aliases`.

Для создания такого списка рассылки системный администратор должен ввести псевдоним в глобальный файл `aliases`, а затем создать внешний файл и при помощи команды `chown` сделать его владельцем пользователя, управляющего списком рассылки. Например, файл `aliases` может содержать следующую строку:

```
sabook: :include:/usr/local/mail/lah.readers
```

Файл `lah.readers` должен находиться в локальной файловой системе, а не в системе, смонтированной средствами NFS[14]. Кроме того, запись в этот файл должна быть разрешена только его владельцу. Для полноты следует также создать псевдоним, соответствующий владельцу списка рассылки, чтобы сообщения об ошибках (“рикошеты”) посылались владельцу, а не отправителю сообщения:

```
owner-sabook: evi
```

О списках рассылки, а также их взаимосвязи с файлом `aliases` пойдет речь в разделе 18.5.

Направление почты в файл

Если объект, на который ссылается псевдоним, — полное имя файла (заключенное в двойные кавычки при наличии специальных символов), то сообщения добавляются в конец указанного файла. Сам файл должен существовать. Вот пример такого псевдонима:

```
complaints: /dev/null
```

Возможность посылать почту в файл или на вход программы очень полезна, но она ослабляет безопасность и потому должна быть ограничена. Приведенный синтаксис допустим только в файле `aliases` и пользовательском файле `.forward` (а также в файлах, которые присоединяются к ним посредством директивы `:include:`). Имя файла не трактуется как адрес, поэтому почта, направленная по адресу `/etc/passwd@host.domain`, будет возвращена.

Некоторые пользовательские агенты позволяют сохранять почту в локальном файле (например, в папке исходящей почты), однако такое сообщение “принадлежит” пользовательскому агенту и никогда не обрабатывается почтовой системой.

Если ссылка на внешний файл содержится в файле `aliases`, то адресуемый файл должен быть полностью открыт для записи (что нежелательно), либо иметь установленный бит смены идентификатора пользователя (setuid) и быть неисполняемым, либо принадлежать основному пользователю программы `sendmail`. Этот пользователь задается с помощью опции `DefaultUser`. Обычно им является пользователь `mailnull`, `sendmail`, `daemon` или пользователь со значениями UID и GID, равными 1.

[14] Если сетевая файловая система смонтирована с флагом `hard` и сервер NFS выйдет из строя, программа `sendmail` “зависнет”. При этом останутся открытыми дескрипторы отдельных файлов и окажутся заблокированными некоторые процессы. В итоге можно превысить лимит на число процессов или открытых файлов, после чего придется перезагружать компьютер.

Если же ссылка на файл содержится в файле .forward, то адресуемый файл должен принадлежать и быть доступным для записи основному получателю сообщения. Необходимо, чтобы у этого пользователя была своя запись в файле /etc/passwd, а его интерпретатор команд был упомянут в файле /etc/shells. Для файлов, принадлежащих пользователю root, следует задать режим доступа 4644 или 4600, т.е. установить бит setuid и отменить возможность выполнения.

Направление почты в программу

Благодаря псевдонимам можно организовать пересылку почты на вход заданной программы. Это реализуется с помощью строки примерно следующего вида:

```
autoftp: "|/usr/local/bin/ftpserver"
```

Однако использование такого способа доставки почты создает еще более серьезные бреши в защите, чем направление почты в файл, поэтому данный механизм тоже разрешен только для файлов aliases и .forward и для файлов, которые подключаются к ним посредством директивы :include:, а стандартная конфигурация sendmail требует использовать ограниченную оболочку smrsh[15]. В файле aliases программа вызывается от имени основного пользователя программы sendmail, в остальных случаях — от имени владельца файла .forward или подключаемого файла. Этот владелец должен иметь учетную запись в файле /etc/passwd, а его интерпретатор команд должен быть упомянут в файле /etc/shells.

Перед выполнением команды, осуществляющей прием почты, вызываемая программа меняет свой рабочий каталог на начальный каталог пользователя (или, если он недоступен, на корневой каталог). Раньше по умолчанию выбирался каталог очереди программы sendmail, но некоторые интерпретаторы команд семейства csh не могли получить к нему доступ.

Примеры псевдонимов

Ниже представлен ряд псевдонимов, которые могут понадобиться системному администратору:

```
# Обязательные псевдонимы[16]
postmaster: trouble, evi
postmistress: postmaster
MAILER-DAEMON: postmaster
hostmaster: trent
abuse: postmaster
webmaster: trouble, trent
root: trouble, trent

# Списки рассылки для локальной "проблемной" почты
trouble: :include:/usr/local/mail/trouble.alias
troubletrap: "/usr/local/mail/logs/troublemail"
tmr: troubletrap,:include:/usr/local/mail/tmr.alias
```

[15] Направление почты в программы может вызвать брешь в защите. Оболочка smrsh будет рассмотрена в разделе 18.12.

[16] Это, конечно же, преувеличение. На самом деле нужны только псевдонимы postmaster и MAILER-DAEMON (этого требуют документы RFC), хотя псевдонимы hostmaster, abuse, webmaster тоже полезны.

```
# Для удобства системного администратора
diary: "/usr/local/admin/diary"
info: "|/usr/local/bin/sendinfo"

# Псевдонимы учебных классов, меняющиеся каждый семестр
sa-class: real-sa-class@nag
real-sa-class: :include:/usr/local/adm/sa-class.list
```

Пример построен так, чтобы при возникновении проблем все пользователи университетской сети могли посылать почту по одному псевдониму: `trouble`. Отчеты о проблемах всегда должны направляться соответствующей группе локальных системных администраторов. В частности, нам бы хотелось так организовать почтовые псевдонимы, чтобы:

- сообщения о проблемах всегда поступали в соответствующую локальную группу;
- на всех компьютерах использовался одинаковый файл псевдонимов;
- отдельные группы администраторов контролировали собственные списки рассылки;
- копии всех сообщений о проблемах записывались в локальный журнальный файл каждой группы.

В показанном выше примере эти цели достигнуты: определение псевдонима `trouble` берется из одного и того же файла на каждом компьютере. Почта, посланная по адресам `trouble@anchor` и `trouble@boulder`, попадает в разные места, несмотря на то, что на компьютерах `anchor` и `boulder` используется одинаковый файл `/etc/mail/aliases`.

В пределах каждого факультета сообщения о проблемах обычно обрабатываются на одном компьютере. Например, при наличии адреса

`trouble@`*`главный_компьютер`*

в файле `trouble.alias` на подчиненном компьютере сообщения о проблемах направляются на соответствующий главный компьютер.

Поступившее сообщение посылается псевдониму `tmr` (trouble mail readers), под которым скрываются читатели “проблемной” почты. Псевдоним `tmr` задан таким образом, что сообщение сначала записывается в журнальный файл благодаря псевдониму `troubletrap`, а затем рассылается пользователям, список которых берется из файла на главном компьютере. Включение администраторов-новичков в список `tmr` дает возможность ознакомить их с типичными вопросами по сопровождению системы, с ответами администраторов и с тем льстивым тоном, которым нужно общаться с пользователями.

Псевдоним `sa-class` является двухуровневым, поэтому файл данных, содержащий список студентов, может управляться с одного компьютера: `nag`. Для псевдонима `sabook`, приводившегося в начале раздела 18.4, должен применяться аналогичный подход, чтобы не требовалось копировать подключаемый файл.

Псевдоним `diary` очень удобен для начинающих администраторов, которым требуется документировать свои действия. В файл `diary` системные администраторы могут направлять сообщения о важных событиях в “жизни” компьютера (обновление ОС и аппаратных средств, поломки и т.п.). Однако не помещайте этот файл в ту же файловую систему, что и журнальные файлы. Хакеры могут добиться переполнения файловой системы и тем самым воспрепятствовать записи в журнальные файлы (что позволит им замести следы).

Переадресация почты

Файл `aliases` — это общесистемный конфигурационный файл, который должен вестись администратором. Если же какой-нибудь пользователь захочет организовать перенаправление своей почты (а для доступа к почте в организации не применяется протокол POP или IMAP), он сможет сделать это путем создания файла `.forward` в своем начальном каталоге. Программа `sendmail` всегда просматривает начальный каталог пользователя на предмет наличия в нем файла `.forward`, если только в переменной `ForwardPath` не задан другой путь к этому файлу. Файл `.forward` удобно применять в случае, когда пользователь хочет читать почту на конкретном компьютере или переходит из одного подразделения в другое и хочет, чтобы почта пересылалась туда.

Файл `.forward` состоит из одной строки, в которой содержится список разделенных запятыми адресов, или из нескольких строк с адресами в каждой. Например:

```
evi@ipn.caida.org
evi@atrust.com
```

или

```
\mcbryan, "/home/mcbryan/archive", mcbryan@f1supi1.gmd.de
```

В первом примере почта пользователя `evi`, вместо того чтобы сохраняться на локальном компьютере, будет пересылаться на компьютер `ipn` в организации CAIDA (Сан-Диего) и на хост xor.com. Второй пример — это запись пользователя, который не доверяет почтовым системам и хочет, чтобы его почта копировалась в три места: в обычный почтовый каталог на локальном компьютере, в постоянный архив всей входящей почты и по временному адресу в Германии, где в настоящей момент находится пользователь. Обратная косая черта перед именем пользователя предписывает сохранять почту на локальном компьютере вне зависимости от установок файлов `aliases` и `.forward`.

Если речь идет о временных изменениях в системе маршрутизации почты, то лучше использовать файл `.forward`, а не глобальный файл `aliases`. Затраты времени (машинного и пользовательского) на изменение общесистемных псевдонимов довольно высоки.

Пользовательский файл `.forward` должен принадлежать пользователю и быть недоступным для записи группе и остальным пользователям. Если программа `sendmail` решит, что путь к файлу `.forward` является безопасным (т.е. на пути от корневого каталога не нарушаются права доступа), то файл `.forward` будет включен в почтовую систему. В противном случае присоединения не произойдет, так как подозрительные файлы `.forward` игнорируются. Права доступа к родительскому каталогу также должны быть правильными (т.е. право записи должен иметь только владелец файлов).

Естественно, программа `sendmail` должна иметь доступ к начальному каталогу пользователя на компьютере, куда доставляется почта, ведь программе нужно определить, присутствует ли там файл `.forward`. Если изменение адреса носит постоянный характер, его необходимо внести в файл `/etc/mail/aliases`, так как начальный каталог и файлы пользователя будут в итоге удалены.

Программа `sendmail` имеет прекрасное средство `FEATURE('redirect')`, которое позволяет обрабатывать изменения почтовых адресов. Если псевдоним пользователя указывает на адрес *пользователь@новый_адрес*.REDIRECT, то почта возвращается отправителю с уведомлением о новом адресе. Поскольку сообщение не пересылается по новому адресу, отправитель вынужден внести изменение в свою адресную книгу и повторно послать сообщение.

Программу sendmail можно настроить на поддержку центрального каталога для файлов .forward. Это, однако, удобнее для программы sendmail, а не для пользователей. Местонахождение файлов .forward задается опцией ForwardPath, которая обычно указывает сначала на центральный каталог, а затем на начальный каталог пользователя. В доменном файле generic.m4, пример которого показан в разделе 18.8, демонстрируется задание центрального каталога для файлов .forward.

Записи глобального файла aliases имеют приоритет над записями файла .forward. А поскольку эти файлы ведутся разными людьми, следует действовать очень аккуратно и следить за тем, чтобы не возникали "петли". Если пользователь сети имеет почтовый каталог (и, следовательно, запись в глобальном файле aliases), он не может посредством файла .forward перенаправлять почту на компьютер, где используется такой же файл псевдонимов. Например, в университете штата Колорадо, где мы применяем глобальный файл aliases, запись

```
evi: evi@boulder
```

и файл .forward на компьютере boulder, содержащий строку

```
evi@anchor.cs
```

образуют "петлю". Почта, адресованная пользователю evi, будет направляться на компьютер boulder, где в файле .forward задана ее пересылка на компьютер anchor в поддомене "cs". Однако в файле aliases на компьютере anchor содержится указание пересылать почту обратно на компьютер boulder... После 25 таких переходов почта будет возвращена отправителю.

Если электронная почта является основным средством коммуникации, то уведомить пользователя о возникновении "петли" непросто. Когда псевдоним задан в виде *\пользователь*[17], почта будет доставляться на локальный компьютер независимо от содержимого общесистемного файла aliases и пользовательского файла .forward. Если именно на этом компьютере пользователь ожидает получить почту — прекрасно, если нет, отправьте сообщение о возникновении "петли" псевдониму postmaster или обратитесь к проверенному средству: телефону.

Хешированная база данных псевдонимов

Поскольку записи файла aliases не упорядочены, прямой поиск в данном файле был бы для программы sendmail неэффективным. По этой причине создается хешированная версия файла aliases. Это делается средствами СУБД Berkeley DB. Хеширование значительно ускоряет поиск псевдонимов, особенно в больших файлах.

Файлы, образованные из файла /etc/mail/aliases, называются: в случае DB — aliases.db, в случае ndbm — aliases.dir и aliases.pag. При каждом изменении файла aliases хешированную базу данных следует перестраивать с помощью команды newaliases, которая просто запускает программу sendmail с соответствующими флагами командной строки (-bi). Если команда newaliases вызывается в автоматическом режиме, сохраняйте выводимые ею сообщения об ошибках, поскольку могут возникнуть проблемы с форматированием.

[17] Возможно, придется использовать не один, а два или более символов обратной косой черты, чтобы один из них "прошел" интерпретатор команд и попал в программу sendmail.

18.5. Списки рассылки и программы для работы с ними

Список рассылки — это гигантский псевдоним, при помощи которого копия каждого адресованного ему сообщения передается каждому члену списка. Списки рассылки напоминают давно существующие телеконференции Usenet, в которых новости распространяются посредством электронной почты.

Списки рассылки обычно объявляются в файле `aliases`, но формируются во внешнем файле. Существуют стандартные соглашения по присвоению имен, понятные программе `sendmail` и специальным программам ведения таких списков. Опытные пользователи тоже придерживаются их. Эти соглашения иллюстрируются следующими примерами:

```
mylist: :include:/etc/mail/include/mylist
owner-mylist: mylist-request
mylist-request: evi
owner-owner: postmaster
```

В данном случае `mylist` — это имя списка рассылки; в файле `/etc/mail/include/mylist` содержится список его членов. Суффикс "-request" и префикс "owner-" довольно распространены. Они указывают на то, что сообщение будет получено координаторами списка. Сообщения об ошибках, возникающие при отправке почты по списку рассылки, и запросы на включение в список новых членов посылаются его владельцу (пользователю `evi`). Косвенная адресация "владелец — запрос — пользователь" удобна, так как адрес владельца (`mylist-request`) указывается в заголовке "Return-Path" каждого сообщения, посылаемого в список рассылки. Выражение "mylist-request" — это более подходящее содержимое заголовка, чем настоящее имя владельца. Сообщения об ошибках, возникающие при отправке писем псевдониму `owner-mylist` (на самом деле это пользователь `evi`), направляются псевдониму `owner-owner`.

Если сообщение не удалось доставить по назначению и владельцу списка возвращается сообщение об ошибке, это называется *рикошетом* (bounce). Если же невозможно доставить само сообщение об ошибке, тогда уведомление об этом называют *двойным рикошетом*. В нашем примере такие уведомления посылаются псевдониму `owner-owner` (т.е. `postmaster`).

Когда используется общесистемный файл псевдонимов, необходимо ввести еще один уровень косвенной адресации, связав псевдоним `mylist` с адресом *реальное_имя_списка@главный_компьютер*, чтобы файл, содержащий список членов, существовал только в одном месте.

Программные пакеты для поддержки списков рассылки

Существует несколько программ, которые автоматизируют работу со списками рассылки. С их помощью пользователи могут добавлять и удалять себя из списка, просматривать информацию о списке, получать файлы по электронной почте и т.д. Среди этих программ отметим следующие:

- Majordomo (`www.greatcircle.com`) — входит в состав SUSE;
- Mailman (`www.list.org`);
- ListProc (`www.cren.net`);

- SmartList (ftp.informatik.rwth.aachen.de);
- listmanager (www.listmanager.org);
- LISTSERV Lite (www.lsoft.com) — свободно распространяемая версия коммерческого пакета LISTSERV.

Если говорить в общем, то SmartList — самая простая и маленькая по объему программа из данного перечня, а ListProc — самая большая и сложная. Остальные — где-то посредине. Все программы отличаются методами управления списками рассылки. Одни больше ориентированы на администраторов (ListProc), другие — на пользователей, управляющих списками рассылки (Majordomo, Mailman, SmartList, LISTSERV Lite). Программы Majordomo, Mailman, listmanager и LISTSERV Lite поддерживают дистанционное администрирование. Администратору списка даже не нужно иметь учетную запись на том компьютере, где находится список, поскольку все операции осуществляются по электронной почте. Большинство программ позволяет создавать информационные подборки (дайджесты) на основе материалов, публикуемых в списке рассылки. Иногда это делается автоматически (ListProc, Mailman, listmanager, LISTSERV Lite), а иногда требуется ручное конфигурирование (SmartList, Majordomo).

Мы предпочитаем пользоваться программой Mailman. Она проста в администрировании и предоставляет владельцам списков широкие возможности в плане управления ими. Программы ListProc и LISTSERV Lite скованы рамками лицензионных соглашений, поэтому менее предпочтительны: первая — дорогая, а вторая — доступна только в исполняемых кодах и обладает усеченными возможностями по сравнению со своим коммерческим вариантом. Мы не пробовали работать с пакетом SmartList, однако нам нравится программа procmail, на основе которой он создан.

Ниже кратко описана каждая из этих программ. За получением более подробной информации обращайтесь к документации.

Majordomo

Majordomo — написанный на языках Perl и C пакет, который можно получить на Web-сайте www.greatcircle.com. Разработка программы прекращена. Версия под номером 2 полностью отличается от исходной версии; однако, похоже, что она в 2004 году была признана неудавшейся, поэтому мы будем описывать только исходную версию, которая используется до настоящего времени. Среди наших демонстрационных дистрибутивов только SUSE включает пакет Majordomo. Он находится в каталоге /usr/lib/majordomo, хотя на man-странице упоминается другой каталог (/usr/lib/mail/majordomo).

Программа Majordomo выполняется с правами непривилегированного пользователя, обычно от имени пользователя majordom или mdom и группы daemon. Поскольку в Linux поддерживаются длинные имена (более 8 символов), в качестве регистрационного имени можно задать majordomo. Пользователь должен принадлежать к числу "доверенных" пользователей программы sendmail, т.е. быть соответствующим образом обозначенным в ее конфигурационном файле (обычно посредством опции confTRUSTED_USERS)[18].

🕮 Дополнительная информация о "доверенных" пользователях приведена в разделе 18.13.

Программа Majordomo конфигурируется с помощью файла majordomo.cf. Этот файл содержит команды на языке Perl, посредством которых инициализируются пере-

[18] "Доверенному" пользователю разрешается менять заголовок "From" сообщения и перестраивать файл aliases.

менные, задаются каталоги, определяются поддерживаемые списки рассылки и правила доставки сообщений об ошибках. Вспомогательная утилита `config-test` позволяет протестировать конфигурационный файл на предмет отсутствия нужных переменных и синтаксических ошибок. В SUSE конфигурационный файл помещается в каталог `/etc`, а утилита `config-test` остается в дистрибутивном каталоге `/usr/lib/majordomo`.

Majordomo требует наличия специальных псевдонимов в файле `aliases` программы `sendmail`. Самый правильный способ их интеграции в систему — создание отдельного файла псевдонимов для Majordomo (последние версии программы `sendmail` поддерживают несколько файлов псевдонимов). Такой файл должен содержать псевдонимы для самой программы Majordomo и набор псевдонимов для каждого управляемого ею списка рассылки. В дистрибутив программы включен файл `majordomo.aliases`, который может служить образцом.

Пользователи часто задают вопрос: как исключить себя из списка рассылки? Что касается списков, управляемых программой Majordomo, то ответ звучит так: чтобы отказаться от получения информации из списка *имя_списка@имя_хоста*, следует направить по адресу majordomo@*имя_хоста* сообщение, в теле (а не в строке "Subject") которого содержится фраза "unsubscribe *имя_списка*" или "unsubscribe *имя_списка адрес_электронной_почты*".

В первом случае сообщение должно быть отправлено с того компьютера, с которого ранее была осуществлена подписка на рассылку. Во втором случае имя компьютера, для которого следует отказаться от подписки, является частью почтового адреса. Просмотрите еще раз примеры, приведенные в разделе 18.2; там рассказывалось, как извлечь необходимую информацию из заголовков сообщения, чтобы отказаться от подписки даже в том случае, когда вы забыли, с какого компьютера когда-то подписались на рассылку. Для некоторых списков рассылки достаточно послать сообщение с текстом "unsubscribe" по адресу *имя_списка*-request@*имя_хоста*.

Никогда не посылайте сообщение об отказе от подписки непосредственно в список рассылки, поскольку оно будет разослано всем его подписчикам и все поймут, что вы "чайник".

Mailman

Mailman — относительно новая программа работы со списками рассылки (версия 2.1.8 появилась в апреле 2006 г.). Ее можно найти на Web-сайте `www.list.org` или в GNU-архивах. Как и Majordomo, программа Mailman написана на языке сценариев, только в данном случае это Python (`www.python.org`).

Толчком к созданию программы Mailman послужили недостатки программы Majordomo: плохое управление "рикошетами", сложный способ конфигурирования дополнительных возможностей (в частности, информационных дайджестов и списков с централизованным управлением), а также низкая производительность при работе с большими списками. В дистрибутив Mailman входит сценарий, позволяющий импортировать списки Majordomo. Программа Mailman располагает также средствами обнаружения спама и борьбы с ним.

Большим преимуществом программы является ее Web-интерфейс. Его наличие упрощает координатору списка (модератору) или почтмейстеру управление списком, а пользователям — подписку, отмену подписки и настройку списка.

ListProc

ListProc — одна из старейших программ управления списками рассылки. Она была написана в 1991 г. Анастасиосом Котсиконасом (Anastasios Kotsikonas) и поддерживалась приблизительно до 1994 г. Затем, после нескольких лет забвения, появилась новая бета-версия программы (1998 г.). Ее можно было бесплатно получить на факультете вычислительной техники в университете Бостона, но на довольно странных условиях лицензирования. Сейчас программа ListProc доступна на Web-сайте `www.cren.net`, однако условия ее получения стали вообще неприемлемыми ($2000 за копию, даже для университетов). Так что забудьте о ListProc и пользуйтесь свободно распространяемым программным обеспечением.

SmartList

Программу SmartList написал Стефан ван ден Берг (Stephen van den Berg), который является также автором утилиты `procmail`. Дистрибутив SmartList можно найти по адресу `www.procmail.org`. Поскольку программа использует утилиту `procmail`, потребуется загрузить файлы `procmail.tar.gz` и `SmartList.tar.gz`. А еще лучше — поискать соответствующий Linux-пакет.

SmartList — простая, небольшая по объему программа. Она представляет собой комбинацию кода на языке C, правил утилиты `procmail` и сценариев интерпретатора команд. С "рикошетами", главной неприятностью списков рассылки, программа борется автоматически. Пользователь автоматически удаляется из списка после того, как направленная в его адрес почта возвращается определенное число раз. Программа SmartList требует наличия в файле `passwd` учетной записи (`smart` или, может быть, `list`), соответствующей "доверенному" пользователю программы `sendmail`.

Дистрибутив SmartList включает специальный редактор `led`, который защищает программу от использования с несогласованным или частично отредактированным файлом конфигурации.

Listmanager

Программа `listmanager` написана на языке C, поэтому работает быстрее программ, написанных на таких языках сценариев, как Perl или Python. Повышению производительности способствует также использование СУБД Berkeley DB вместо обычных файлов. Возможности программы велики; у нее есть Web-интерфейс, поддерживаются дайджесты, улучшена безопасность и т.д.

В целом программа `listmanager` выглядит довольно многообещающе. Единственным ее недостатком является недоступность исходного кода. Согласно информации, приведенной на Web-странице `www.listmanager.org`, автор в настоящее время подготавливает исходный код к публикации. Пока что в нашем распоряжении имеется исполняемый файл для Linux.

LISTSERV Lite

LISTSERV Lite — усеченная версия пакета LISTSERV, коммерческого продукта компании L-Soft International, Inc. Не все возможности полнофункциональной версии реализованы, поэтому программа способна управлять всего десятью списками рассылки по 500 членов каждый. Программа работает с правами псевдопользователя `listserv`, который должен быть владельцем ее файлов. Желательно также создать группу `listserv`.

Программа снабжена Web-интерфейсом, посредством которого можно как подписываться на рассылку, так и управлять самой программой.

Дистрибутив LISTSERV Lite можно загрузить с Web-сайта `www.lsoft.com`. Исходный код программы не распространяется, однако доступны предварительно скомпилированные исполняемые файлы для многих версий UNIX и Linux. Тем, кто уже работал с пакетом и располагает созданными им списками рассылки, имеет смысл использовать усеченную его версию. Всем остальным рекомендуем остановить свой выбор на одной из описанных ранее программ с открытым исходным кодом.

LDAP

LDAP (Lightweight Directory Access Protocol — упрощенный протокол доступа к каталогам) обеспечивает доступ к унифицированной административной службе каталогов. Этот протокол существует уже не один год, но лишь недавно начал приобретать популярность.

Администраторы обнаружили, что протокол LDAP очень удобен для решения самых разнообразных задач, а именно:

- конфигурирования программы `sendmail` (псевдонимы, виртуальные домены, почтовые каталоги, базы доступа, таблицы);
- управления пользователями (регистрационные имена, пароли и т.д.);
- ведения административных конфигурационных файлов (например, в SUSE);
- в качестве замены NIS;
- для использования с модулями PAM.

Предполагается, что со временем LDAP превратится в универсальный инструмент работы с каталогами. К сожалению, все еще отсутствуют утилиты, предназначенные для автоматизации типичных административных задач средствами LDAP.

В основе LDAP лежат протоколы ISO и служба каталогов X.500, а такие родственные связи предполагают сложность, громоздкость, множество недостатков и т.д. Однако наличие в названии протокола буквы “L” (lightweight, т.е. “упрощенный”) указывает на отсутствие таковых. Версии 1 и 2 протокола уже прошли процесс стандартизации. Работа над версией 3 также завершена. Все версии обладают обратной совместимостью. Особенностью версии 3 является поддержка иерархических серверов. Это означает, что в ответ на запрос сервер может вернуть ссылку на другой сервер. В версии 2 поддерживается только иерархическая организация данных.

Протокол LDAP очень удобен для работы с почтовыми псевдонимами, особенно сейчас, когда в программе `sendmail` реализована внутренняя поддержка протокола. Программа `sendmail` может посылать серверу LDAP запрос на поиск псевдонимов, вместо того чтобы осуществлять поиск непосредственно. Протокол управляет также маршрутизацией почты и виртуальными доменами. Средства поддержки LDAP должны быть встроены в исполняемый код программы `sendmail` на этапе компиляции.

Тем, кто ищут программную реализацию протокола LDAP, можем порекомендовать пакет OpenLDAP (`www.openldap.org`). Организация OpenLDAP получила в свое распоряжение, а затем улучшила код предыдущего сервера LDAP, разработанного в университете штата Мичиган. Дополнительная информация о программном обеспечении LDAP приводилась в разделе 17.5.

Записи базы данных LDAP напоминают элементы файла `termcap`, но с более длинными именами переменных. Атрибуты (имена переменных) в базе данных LDAP еще

не полностью стандартизированы, что может привести к несовместимости реализаций протокола.

Атрибуты, находящиеся в первой строке записи, определяются в файле конфигурации LDAP. В приводимых ниже примерах предполагается, что серверный демон LDAP (`slapd` в случае пакета OpenLDAP) сконфигурирован с корневым отличительным именем (запись `rootdn` — "root distinguished name") следующего вида:

```
"cn=root,dc=synac,dc=net"
```

Атрибут `dc` (domain component) указан дважды, поскольку доменный компонент имени не может включать точку; следовательно, для представления домена synac.net требуются два элемента. Последующие атрибуты (имена переменных) могут быть произвольными. Регистр символов в данном случае не важен. Программа `sendmail` (которая ищет предопределенные имена атрибутов, интерпретируя их заданным образом), сервер LDAP и утилита создания базы данных LDAP должны придерживаться единого соглашения об именовании атрибутов.

Среди допустимых атрибутов, которые могут находиться в первой строке записи (ключи базы данных), упомянем `dn` (доменное имя), `dc` (доменный компонент), `o` (название организации), `c` (название страны) и `uid` (уникальный идентификатор, например регистрационное имя).

`sendmail` использует сервер LDAP подобно тому, как она использует сервер имен DNS. Она вызывает сервер DNS для превращения имен в IP-адреса, что необходимо для отправки сообщения. Она вызывает сервер LDAP для поиска псевдонимов, чтобы можно было доставить сообщения в нужное место. В обоих случаях поиски производятся от плоских файлов (`hosts` и `aliases`) до баз данных, при этом запросами управляют серверы.

По умолчанию программа `sendmail` распознает следующие информационные дескрипторы LDAP:

```
mailLocalAddress
mailRoutingAddress
mailHost
```

В `sendmail` версии 8.12 этот набор значительно расширен (см. файл `cf/sendmail.schema`) и разрешается определять пользовательские схемы с произвольными дескрипторами.

Ниже приведен пример файла `ldap.conf` для демона `slapd`:

```
# Файл ldap.conf должен быть доступен для чтения всем пользователям.
#
BASE   dc=synack, dc=net
HOST   gw.synack.net
PORT   389
```

Этот файл обеспечивает поддержку базы данных с записями следующего вида:

```
dn: uid=jon, dc=synack, dc=net
objectClass: inetLocalMailRecipient
mailLocalAddress: jon@synack.net
mailRoutingAddress: stabilej@cs.colorado.edu
uid: jon
```

Если в конфигурационном файле программы `sendmail` присутствует выражение `FEATURE(ldap_routing)`, то адрес получателя входящего сообщения сравнивается с содержимым поля `mailLocalAddress`. В случае совпадения письмо пересылается по адре-

су, указанному в поле mailRoutingAddress. Наличие строки objectClass необходимо: таковы требования документа RFC, который определяет взаимодействие протокола LDAP и почтовой системы.

На хосте gw.synack.net приведенный фрагмент базы данных соответствует псевдониму

```
jon: stabilej@cs.colorado.edu
```

Все это несколько громоздко, не правда ли? Конечно, такие записи базы данных могли бы заменить обычные элементы файла aliases, применяемые для определения почтовых каталогов пользователей.

В некоторых больших организациях информация об инициализации пользователей (например, предпочтительный почтовый ящик пользователя) уже хранится в другой базе данных. Если у вас сделано точно так же, то, возможно, вам придется написать некоторые небольшие сценарии для извлечения этой информации и передачи ее в LDAP. Если ваша организация небольшая, то лучше работать с традиционным файлом /etc/mail/aliases, или написать сценарий, который будет генерировать правильные записи LDAP из командной строки.

Кроме того, файл aliases все еще остается лучшим средством объявления списков рассылки (при помощи директивы :include:). Программы управления списками обычно передают (через канал) сообщение сценарию-упаковщику, после чего повторно посылают сообщение. LDAP-запрос может вернуть псевдонимы, указывающие на файлы :include, или на программы, такие как majordomo, но не может непосредственно вызвать программу.

Программа sendmail версии 8.12 позволяет хранить в базе данных LDAP дополнительную информацию (например, таблицы и классы). Подробнее о настройке протокола LDAP в sendmail будет рассказываться в разделе 18.9.

18.6. "Почтмейстер" sendmail

Во всех пяти наших демонстрационных дистрибутивах Linux имеется почтовый транспортный агент. В Red Hat, Fedora и SUSE это sendmail, а в Debian и Ubuntu — Exim. Последний будет вкратце описан в разделе 18.15. Многие его конструкции, а также конфигурационные элементы имеют аналоги в sendmail. По причине того что эта глава и так слишком велика, мы решили подробно описывать только программу sendmail.

Программа sendmail способна адаптироваться к многообразию различных стандартов благодаря гибкости своего конфигурационного файла, который позволяет программе удовлетворять потребностям самых разных пользователей. Поэтому остальная часть главы посвящена анализу структуры этого файла, называющегося sendmail.cf.

Программа sendmail является транспортным агентом, связкой между пользовательскими агентами и агентами доставки. Она работает по протоколу SMTP и доставляет сообщения на удаленные компьютеры через Internet. Круг ее обязанностей таков:

- управление сообщениями после того, как они сгенерированы пользователями;
- анализ адресов получателей;
- выбор соответствующего агента доставки или транспортного агента;
- преобразование адресов в форму, понятную агенту доставки;
- переформатирование заголовков;
- передача преобразованных сообщений агенту доставки.

Кроме того, программа генерирует сообщения об ошибках и возвращает сообщения отправителю, если их оказалось невозможно доставить.

История программы sendmail

На момент подготовки этой книги в большинстве дистрибутивов Linux содержалась одна из версий программы sendmail семейства V8. Обычно эти версии на один-два выпуска отстают от версии, выпускаемой компанией Sendmail, Inc. Разработчики дистрибутивов чаще всего просто настраивают ту или иную версию под свою систему, а затем отказываются обновлять базовый вариант системы с учетом текущих версий программы.

Мы опишем программу sendmail версии 8.13, оставив в стороне версии 5 и IDA, которые устарели. В версии 8 используется макропроцессор m4, упрощающий создание файла конфигурации для стандартных ситуаций (а в большинстве случаев больше ничего и не требуется).

К сожалению, при возникновении проблем приходится заниматься отладкой низкоуровневого файла конфигурации. Нам доводилось слышать о нем такие нелестные отзывы, как "непостижимый", "устрашающий", "крайне тяжелый", "загадочный", "трудный для понимания", "мерзкий", "утомляющий", "издевательский", "запутанный", "громоздкий", "нелепый", "сбивающий с толку", "закрученный" и еще много чего. В ранних изданиях книги описанию этого файла уделялось много внимания, но поскольку нынешняя его важность с точки зрения администрирования невелика, мы предлагаем читателям обратиться к руководству *Sendmail Installation and Operations Guide*, поставляемому вместе с программой.

Появление новых выпусков программы sendmail зачастую вызвано проблемами с безопасностью. В связи с этим мы рекомендуем читателям просмотреть раздел "Release Notes" на сервере www.sendmail.org и загрузить необходимые "заплаты". Пакеты sendmail можно обновить прямо у поставщика вашей системы Linux; вам нужно будет лишь проследить за тем, чтобы полученные вами бинарные файлы соответствовали текущей версии sendmail. Если вам понадобится скомпилировать и инсталлировать исходный дистрибутив прямо с сайта www.sendmail.org, вам потребуются компилятор языка C и препроцессор m4 (они обычно входят в дистрибутивы Linux).

Иногда трудно определить, какая версия программы sendmail установлена в системе. Попробуйте выполнить команду

```
# /usr/sbin/sendmail -d0.1 -bt < /dev/null
```

и программа сообщит свою версию, параметры компиляции и свои идентификационные имена, найденные после чтения файла конфигурации. Флаг -d задает уровень отладки (об этом пойдет речь в разделе 18.14), флаг -bt переводит программу в режим тестирования адреса, а чтение входного потока из устройства /dev/null приводит к тому, что адрес для тестирования будет отсутствовать. Результат работы команды в Red Hat будет примерно таким:

```
Version 8.11.6
 Compiled with: LDAPMAP MAP_REGEX LOG MATCHGECOS MIME7TO8
    MIME8TO7 NAMED_BIND NETINET NETINET6 NETUNIX NEWDB NIS
    QUEUE SASL SCANF SMTP TCPWRAPPERS USERDB
============= SYSTEM IDENTITY (after readcf) ==========
       (short domain name) $w = coyote
   (canonical domain name) $j = coyote.toadranch.com
         (subdomain name) $m = toadranch.com
              (node name) $k = coyote.toadranch.com
========================================================
```

Программа sendmail должна использовать записи MX базы данных DNS, поэтому ее всегда следует компилировать с установленной опцией NAMED_BIND (как показано в примере). Однобуквенные переменные вроде $w берутся из низкоуровневого файла конфигурации либо определяются во время выполнения программы.

Инсталляция программы sendmail

Принципы инсталляции программы sendmail версии 8.12 немного изменились. Больше нет необходимости устанавливать бит setuid, чтобы программа запускалась с правами пользователя root. Вместо этого устанавливается бит setgid для смены группы на smmsp. Эта группа, а также одноименный пользователь должны быть созданы до начала инсталляции (загадочное имя расшифровывается как "**s**end**m**ail **m**ail **s**ubmission **p**rogram"). Мы присваиваем им значения UID и GID, равные 25, что соответствует номеру порта протокола SMTP. В файле /etc/passwd пользователю smmsp должна быть назначена одноименная группа. Наличие специальных пользователя и группы позволяет программе sendmail отказаться от особых привилегий, что повышает безопасность системы.

Второе, важное с точки зрения администратора, изменение заключается в том, что теперь у программы sendmail два конфигурационных файла: submit.cf и sendmail.cf. Первый из них задает параметры обработки почты, создаваемой на локальном компьютере (и поступающей в почтовую систему), а второй управляет входящей почтой и почтой, находящейся в очереди. Файл submit.cf является частью дистрибутива и одинаков в любой системе; менять его обычно не приходится.

В этом подразделе мы кратко опишем процесс инсталляции программы sendmail. Более детальную информацию, касающуюся, в том числе, конкретных платформ и операционных систем, можно найти в дистрибутиве программы. В следующем подразделе речь пойдет об инсталляции в Debian при помощи команды apt-get. Если программа уже инсталлирована в системе и лишь обновляется, то может оказаться, что некоторые этапы (например, установка файлов справки) повторять не нужно.

Для работы программы требуются:

- исполняемый файл sendmail, который обычно инсталлируется в каталог /usr/sbin;

```
-r-xr-sr-x   root    smmsp    /usr/sbin/sendmail
```

- конфигурационные файлы /etc/mail/sendmail.cf и (в версии 8.12 и выше) /etc/mail/submit.cf, устанавливаемые системным администратором;

```
-rw-r--r--   root    bin      /etc/mail/sendmail.cf
-rw-r--r--   root    bin      /etc/mail/submit.cf
```

- каталоги почтовых очередей /var/spool/mqueue и (в версии 8.12 и выше) /var/spool/clientmqueue, создаваемые системным администратором или инсталляционной утилитой;

```
drwxrwx---   smmsp   smmsp    /var/spool/clientmqueue
drwx------   root    wheel    /var/spool/mqueue
```

- ссылки на программу sendmail (newaliases, mailq, hoststat и др.)[19];
- безопасные локальные агенты доставки smrsh и mail.local (обычно инсталлируются в каталог /usr/libexec).

[19] Будьте внимательны. В некоторых дистрибутивах используются жесткие ссылки, поэтому после обновления программы может оказаться, что версия файла sendmail не соответствует версии файла newaliases, и это станет причиной возникновения трудно обнаруживаемых ошибок.

Последнюю версию программы можно загрузить с хоста www.sendmail.org. При компиляции и последующей инсталляции пакета следуйте инструкциям, приведенным в файле INSTALL. Начните с создания пользователя и группы smmsp. Не назначайте этому пользователю реальный интерпретатор команд. Вот типичная запись файла /etc/passwd:

```
smmsp:x:25:25:Sendmail Mail Submission Prog:/nonexistent:/bin/false
```

А вот соответствующая запись файла /etc/group:

```
smmsp:*:25:smmsp
```

Чтобы скомпилировать программу, перейдите в каталог sendmail дистрибутива, запустите сценарий Build и выполните команду Build install. В файле devtools/OS/Linux содержатся установки, на основании которых осуществляется компиляция программы в среде Linux. В различных дистрибутивах Linux многие элементы не стандартизированы, поэтому установки указанного файла в ряде случаев выбраны по принципу "наименьшего общего знаменателя" и могут не вполне соответствовать особенностям конкретной системы.

И еще один важный момент: до начала компиляции нужно определиться в отношении формата базы данных, а также стратегии взаимодействия программы с административными СУБД, такими как, например, NIS. Для баз данных, хранящихся на диске, мы рекомендуем СУБД Berkeley DB, которая в файле Makefile задается как NEWDB (это установка по умолчанию).

Только не редактируйте файл Makefile вручную. Создайте собственный файл site.config.m4, посредством которого будет осуществляться настройка программы sendmail на работу в локальной среде, и поместите его в каталог devtools/Site. Например, если планируется использовать протокол LDAP и новую библиотеку фильтрации почты, создайте файл site.config.m4 со следующими строками:

```
APPENDDEF(`confMAPDEF', `-DLDAPMAP')
APPENDDEF(`confLIBS', `-lldap -llber')
APPENDDEF(`conf_sendmail_ENVDEF', `-DMILTER')
```

Если макрос define заменяет текущее определение атрибута, то макрос APPENDDEF расширяет существующее определение.

Установки, касающиеся локальной системы, автоматически подключаются при компиляции программы sendmail с помощью следующей команды:

```
# sh ./Build
```

Для инсталляции программы в соответствующие каталоги выполните такую команду:

```
% sudo sh ./Build install
```

Программа sendmail обычно не должна работать под управлением демона inetd/xinetd, поэтому ее следует явно вызывать из сценариев запуска системы. Типичный фрагмент сценария выглядит так:

```
if [-f /usr/sbin/sendmail -a -f /etc/mail/sendmail.cf];
then
   (cd /var/spool/clientmqueue; rm -f [tTx]f*)
   (cd /var/spool/mqueue; rm -f [tTx]f*)
   /usr/sbin/sendmail -bd -q30m        ### обработчик обычной очереди
   /usr/sbin/sendmail -AC -q30m &      ### обработчик клиентской
                                       ### очереди (в версии 8.12)
   echo -n ' sendmail' > /dev/console
fi
```

В данном случае проверяется наличие исполняемого и конфигурационного файлов, после чего программа запускается в режиме демона. Пакет `sendmail` для вашего дистрибутива Linux должен содержать подходящий сценарий запуска, находящийся в каталоге `/etc/init.d`.

Некоторые пользовательские агенты, вместо того чтобы самостоятельно сделать соответствующий SMTP-запрос, в момент передачи пользовательского сообщения в почтовую систему запускают программу `sendmail` (иногда с флагом `-bm` или `-bs`). В подобной ситуации программа руководствуется установками конфигурационного файла `submit.cf` и помещает сообщения в каталог очереди `/var/spool/clientmqueue`. Если программа вызывается с флагом `-Am` или `-Ac`, то сообщения направляются в каталоги `mqueue` и `clientmqueue`, соответственно.

В Red Hat и Fedora есть сценарий запуска программы (`/etc/init.d/sendmail`), который не удаляет "мусор" из каталогов очередей (как в показанном выше примере), но все же перестраивает таблицы базы данных. В `sendmail` версии 8.12 и выше обработкой таблиц занимается протокол LDAP, поэтому перестраивать таблицы после внесения изменений не нужно. Параметры, заданные в файле `/etc/sysconfig/sendmail`, определяют, должна ли программа запускаться в режиме демона и как часто нужно запускать обработчик очереди. Именно в этом файле следует определять поведение программы при запуске в данной системе. По умолчанию программа запускается в режиме демона, просматривая очередь каждые 30 минут.

В SUSE сценарий запуска (`/etc/init.d/sendmail`) просто проверяет наличие исполняемого и конфигурационного файлов, а затем вызывает программу с аргументами, содержащимися в переменной среды `SENDMAIL_ARGS`. Эти аргументы задаются в файле `/etc/rc.config.d/sendmail.rc.config`. В SUSE программа тоже запускается в режиме демона и просматривает очередь каждые 30 минут.

Приведенный выше фрагмент сценария (или подобный ему) следует добавить в файл `/etc/init.d/sendmail`. Можно также воспользоваться более сложным сценарием из руководства по инсталляции, очищающим очереди сообщений, обработка которых была прервана.

Традиционно файлы, необходимые программе `sendmail`, были распределены по системе и находились в таких каталогах, как `/usr/lib`, `/etc`, `/usr/ucblib` и `/usr/share`. Начиная с версии 8.10 предполагается, что все необходимое (за исключением каталогов очередей) хранится в структуре каталога `/etc/mail`[20]. Надеемся, что поставщики систем поймут этот намек и будут хранить файлы в каком-то одном месте.

Инсталляция программы `sendmail` в Debian и Ubuntu

В Debian для инсталляции программы `sendmail` следует воспользоваться утилитой `apt-get`, которая установит программы `sendmail` и `m4`, попутно деинсталлировав Exim. Далее утилита предложит сконфигурировать программу. В случае положительного ответа запускается сценарий, который задает ряд вопросов (около 20) относительно желаемой конфигурации. Обычно достаточно давать ответы, предлагаемые по умолчанию. Для каждого вопроса приводится

[20] Это условие пока что не всегда выполняется. Например, файл `sendmail.pid` и иногда файл статистики хранятся в другом месте.

стандартный ответ (в квадратных скобках). Единственный вопрос, требующий нестандартного ответа, касается "почтового имени": по умолчанию предлагается короткое имя хоста (например, `lappie`), тогда как необходимо полное имя (например, `lappie.toadranch.com`).

Отвечать на вопросы сценария имеет смысл в том случае, когда вы хорошо знакомы с различными опциями и средствами программы `sendmail` (описываются ниже). Конфигурационный сценарий не учитывает содержимое общесистемного доменного файла, который подключается по умолчанию. В результате может оказаться так, что вы отказались включать средство, а оно все равно включено (например, средство `redirect`).

Файл переключения служб

В Linux имеется специальный конфигурационный файл `/etc/nsswitch.conf`, называемый *файлом переключения служб* (service switch file). В нем перечислены способы выполнения стандартных запросов, таких как поиск компьютеров и пользователей. Этот файл определяет также порядок применения методов поиска в тех случаях, когда для используемого типа запроса указано несколько методов.

📖 Файл переключения служб подробно описывался в главе 17.

Обычно программы не заботятся о наличии файла переключения служб. Однако программа `sendmail` слишком педантична и тщательно контролирует операции поиска, поэтому в настоящее время она игнорирует общесистемный файл, используя свой собственный (`/etc/mail/service.switch`).

В файле переключения служб есть два поля, которые влияют на работу почтовой системы: `aliases` и `hosts`. Возможные значения поля `hosts` таковы: `dns`, `nis`, `nisplus` и `files`. Для поля `aliases` возможны следующие значения: `files`, `nis`, `nisplus` и `ldap`. Поддержка всех механизмов поиска (за исключением файлового) должна быть встроена в программу `sendmail` на этапе компиляции.

Файл `/etc/mail/service.switch` содержит следующие записи:

```
aliases  files nisplus nis          # если программа компилируется
                                    # с поддержкой служб nis/nis+
hosts    dns nisplus nis files
```

Режимы работы

Программа `sendmail` способна работать в нескольких режимах, которые задаются с помощью флага `-b`. Он всегда используется в сочетании с другим флагом, определяющим роль, которую предстоит играть программе. Возможные значения флагов перечислены в табл. 18.4.

Если входящая почта будет поступать из Internet, то программу `sendmail` следует запускать в режиме демона (`-bd`). В этом режиме программа прослушивает сетевой порт 25[21], ожидая поступления запросов. Обычно указывается также флаг `-q`, который задает периодичность обработки очереди почтовых сообщений. Например, флагам `-q30m` и `-q1h` соответствуют интервалы 30 минут и 1 час, соответственно.

[21] Порты, прослушиваемые программой `sendmail`, задаются с помощью макроса `DAEMON_OPTIONS`. Порт 25 выбран по умолчанию.

Таблица 18.4. Флаги командной строки для основных режимов работы программы `sendmail`

Флаг	Назначение
`-bd`	Работа в режиме демона, прослушивание порта 25
`-bD`	Работа в роли демона, но в интерактивном, а не фоновом режиме*
`-bh`	Отображение информации о последнем соединении (то же, что команда `hoststat`)
`-bH`	Удаление с диска старой информации о соединениях (то же, что команда `purgestat`)
`-bi`	Инициализация хешированных псевдонимов (то же, что команда `newaliases`)
`-bm`	Работа в обычном режиме доставки почты (по умолчанию)
`-bp`	Отображение содержимого очереди сообщений (то же, что команда `mailq`)
`-bP`	Отображение числа записей в очередях (в версии 8.12 и выше)
`-bs`	Включение режима SMTP-сервера (чтение данных из стандартного входного потока, а не из порта 25)
`-bt`	Включение режима проверки адресов
`-bv`	Только проверка адресов, почта не отправляется

* Этот режим применяется при отладке, когда требуется увидеть сообщения об ошибках и различные служебные сообщения.

Программа `sendmail` обычно доставляет сообщения немедленно, лишь ненадолго сохраняя их в очереди для надежности. Если же компьютер слишком занят или адресат недоступен, программа ставит сообщение в очередь, а через некоторое время вновь пытается отправить его. Раньше всякий раз, когда программа обрабатывала очередь, она порождала дочерний процесс, но теперь используются постоянные обработчики очередей, запускаемые, как правило, на этапе начальной загрузки системы. В документе RFC1123 рекомендуется обрабатывать очередь с периодичностью 30 минут. Поскольку программа блокирует рабочие файлы, одновременные обращения к очереди не вызывают конфликтов.

В `sendmail` версии 8.12 появилась новая возможность, упрощающая обработку больших списков рассылки и очередей. Это так называемые группы очередей. Они будут описаны в конце раздела 18.14.

Программа `sendmail` читает свой файл конфигурации `sendmail.cf` только в момент запуска, поэтому при изменении этого файла нужно либо уничтожить и перезапустить процесс `sendmail`, либо послать процессу сигнал отбоя (HUP). Программа создает файл `sendmail.pid`, содержащий идентификатор процесса `sendmail` и текст команды, использованной для его запуска. В командной строке программы следует указывать абсолютный путь, так как она перезапускает себя с помощью системного вызова `exec()` в ответ на сигнал отбоя. Наличие файла `sendmail.pid` позволяет уничтожать процесс такой командой:

```
# kill -HUP `head -1 sendmail.pid`
```

Раньше местонахождение PID-файла задавалось в виде параметра компиляции, но сейчас оно может устанавливаться в конфигурационном `mc`-файле при помощи опции `confPID_FILE`:

```
define(confPID_FILE, `/var/run/sendmail.pid')
```

Стандартное значение этой опции зависит от операционной системы, но обычно это /var/run/sendmail.pid либо /etc/mail/sendmail.pid. В Red Hat, Fedora, Debian и Ubuntu применяется первый вариант, а в SUSE файл хранится в каталоге /var/run/sendmail.

Очередь почтовых сообщений

Если компьютер слишком загружен и не может доставить сообщение немедленно или недоступен сам адресат, то сообщение помещается в каталог очереди. Программа sendmail работает в качестве агента подачи почты, прослушивая порт 587, а также играет свою традиционную роль демона, прослушивающего порт 25 и контролирующего каталог очереди /var/spool/mqueue. Некоторые пользовательские агенты (к примеру, mh и /bin/mail) тоже способны взаимодействовать через порт 587, но ряд агентов (Eudora, Outlook и др.) обращается к программе sendmail непосредственно по протоколу SMTP через порт 25. Начиная с версии 8.12 программы подачи почты вставляют сообщения в почтовую систему, используя каталог /var/spool/clientmqueue и руководствуясь установками файла submit.cf.

Программа sendmail позволяет создавать сразу несколько почтовых очередей и объединять очереди в группы. Если, к примеру, каталог mqueue содержит подкаталоги q1, q2 и q3 и каталог очереди задается как /var/spool/mqueue/q*, то будут использоваться все три очереди. Это способствует повышению производительности при высокой загруженности системы[22]. В случае длинного списка рассылки программа sendmail способна разбить список адресов получателей, указанный в конверте сообщения, на несколько списков меньшего размера и закрепить их за разными группами очередей. Подобный прием еще больше повышает производительность, так как появляется возможность обрабатывать несколько списков одновременно.

Группы очередей — это новинка версии 8.12. Они позволяют четко управлять доставкой отдельных типов сообщений. Любой параметр, связанный с очередью, может быть задан и для группы очередей, включая приоритет выполнения (назначается с помощью команды nice). Почтовые сообщения направляются в ту или иную группу в зависимости от адреса самого первого получателя сообщения. Стандартная группа называется mqueue; она определяется автоматически и доступна по умолчанию, не требуя никакого конфигурирования. Подробнее о группах очередей будет рассказываться в конце раздела 18.14.

Когда сообщение ставится в очередь, оно записывается по частям в разные файлы. Имя каждого из этих файлов содержит двухбуквенный префикс, обозначающий фрагмент сообщения, и случайный идентификатор, который образуется из идентификатора процесса sendmail. Идентификатор не является константой, потому что программа sendmail постоянно порождает новые процессы с помощью системного вызова fork. Фрагменты, на которые может быть разбито сообщение, перечислены в табл. 18.5.

Если в каталоге очереди имеются подкаталоги qf, df или xf, то в них будут размещаться соответствующие фрагменты сообщения. Файл qf содержит не только заголовок сообщения, но также адреса конверта, дату возврата сообщения в случае невозможности доставки, приоритет сообщения в очереди и описание причины, по которой оно там находится. Каждая строка начинается с однобуквенного кода, который идентифицирует остальную часть строки.

[22] Каталоги являются эффективным хранилищем данных только тогда, когда не содержат слишком много файлов. Если же почтовый сервер работает с большим числом списков рассылки, то каталог очереди может так переполниться, что скорость поиска в нем станет неприемлемой.

Таблица 18.5. Префиксы файлов сообщений в очереди

Префикс	Содержимое файла
qf	Заголовок сообщения и управляющая информация
df	Тело сообщения
tf	Временная версия файла qf на время его обновления
Tf	Присутствие этого файла означает, что было сделано 32 или более безуспешных попыток блокирования
Qf	Присутствие этого файла означает, что сообщение не удалось доставить вследствие "рикошета"
xf	Временный файл сообщений об ошибках, поступающих от агентов доставки

Для каждого сообщения, которое ставится в очередь, должны существовать файлы qf и df. Все остальные файлы используются программой sendmail при попытках доставить сообщение. В случае перезагрузки компьютера после сбоя необходимо удалять файлы tf, xf и Tf из каждого каталога очереди. Администратор почтовой системы должен периодически проверять наличие файлов Qf, чтобы знать, не приводят ли ошибки локальной конфигурации к возврату сообщений.

Очередь сообщений может стать причиной конфликтов в нескольких ситуациях. Во-первых, существует вероятность переполнения файловой системы (поэтому старайтесь не помещать каталоги /var/spool/mqueue и /var/log в один раздел). Во-вторых, может переполниться сама очередь. Наконец, в очереди могут появиться "осиротевшие" сообщения.

Программа sendmail имеет опцию управления дисковым пространством (confMIN_FREE_BLOCKS). Если файловая система, которая содержит очередь сообщений, превышает допустимый предел заполненности, почта перестает приниматься, и до тех пор, пока не освободится место на диске, будет выдаваться сообщение "try again later" (попробуйте позднее). Это позволяет прекратить прием почты до того, как диск переполнится и система "зависнет".

📖 Подробнее о записях MX базы данных DNS рассказывалось в разделе 15.7.

Когда выходит из строя основной концентратор почты, резервные MX-хосты оказываются под угрозой лавины из тысяч сообщений. В этой ситуации программа sendmail может породить слишком много экземпляров самой себя и тем самым "монополизировать" систему. Существует ряд способов повышения производительности очень загруженных систем; мы опишем их в разделе 18.13. До появления версии 8.12 для очистки переполненной очереди требовалось переместить ее во временный каталог, продолжить обработку новой почты в обычном режиме, а когда все стабилизируется — запустить отдельный экземпляр программы sendmail для обработки переполненной очереди. Вот пример обработки одиночной очереди:

```
# kill `head -1 sendmail.pid`
# mv mqueue cloggedqueue  /* В другую файловую систему, если необходимо */
# mkdir mqueue            /* Задаем владельца и права доступа */
# chown root mqueue
# chmod 700 mqueue
# /usr/sbin/sendmail -bd -q1h &
```

Когда все нормализуется, запустите программу sendmail с такими флагами:

```
# /usr/sbin/sendmail -oQ/var/spool/cloggedqueue -q
```

Эти флаги указывают программе, где находится каталог переполнившейся очереди, и дают ей команду немедленно обработать эту очередь. Повторяйте процедуру до тех пор, пока очередь не очистится. Но следует помнить, что начиная с версии 8.12 программа sendmail использует жесткие ссылки, которые могут перестать работать при перемещении каталога очереди. Таким образом, для обработки переполненных очередей лучше выделить резервный компьютер, на который ссылаются записи MX.

Момент переполнения очереди зависит от размеров почтовой системы и аппаратных средств, на которых работает программа sendmail. Естественно, у обычной рабочей станции и у концентратора почты на хосте aol.com, обрабатывающего миллионы сообщений в день, будут разные критерии, по которым определяется, переполнена очередь или нет. Информация о методике измерения трафика приведена в разделе 18.14.

18.7. Конфигурирование программы sendmail

До появления версии 8.12 действиями программы sendmail управлял единственный файл конфигурации: /etc/mail/sendmail.cf (прежде находился в каталоге /etc или /usr/lib). Теперь у программы есть и второй конфигурационный файл: /etc/mail/submit.cf. Флаги запуска программы определяют выбор файла: в случае флагов -bm, -bs и -bt это submit.cf (если он существует), а во всех остальных режимах — sendmail.cf. Естественно, существуют флаги командной строки и конфигурационные опции, позволяющие менять имена самих файлов, но лучше этого не делать.

В конфигурационном файле определяется следующее:

- правила выбора агентов доставки,
- правила подстановки адресов,
- форматы заголовков почтовых сообщений,
- опции,
- меры безопасности,
- средства защиты от спама.

Формат низкоуровневого файла конфигурации задумывался таким образом, чтобы файл легко поддавался синтаксическому анализу. По этой причине файл мало понятен пользователям, и его сопровождение представляет собой самую серьезную административную задачу в области электронной почты.

Файл конфигурации необходим программе sendmail любой версии. Вместе с тем, в современных версиях процесс конфигурирования несколько упрощен благодаря макросам препроцессора m4, которые позволяют скрыть многие сложные моменты. Если провести аналогию с языками программирования, то можно сказать, что первичный файл конфигурации — это уровень языка ассемблера, а конфигурирование средствами m4 — это уровень языка Perl[23].

[23] Язык конфигурирования программы sendmail является “функционально полным по Тьюрингу”. Это означает, что он может быть использован для написания любой возможной компьютерной программы. Те, кто сталкивались с первичным файлом конфигурации, знают, насколько мощным является применяемый в нем язык.

Когда макросы m4 впервые появились, показалось, что ими можно будет пользоваться в 80–90% случаев. В действительности этот процент оказался еще выше: порядка 98%. Поэтому мы рассмотрим только конфигурирование средствами m4. Вникать в структуру низкоуровневого файла конфигурации приходится лишь в том случае, когда требуется устранить какую-то серьезную проблему или расширить почтовую систему каким-нибудь необычным способом.

Существуют три основных источника, из которых можно почерпнуть подробные сведения о программе sendmail. Это книга *sendmail* издательства O'Reilly, написанная Брайаном Косталесом и Эриком Оллманом, работа Эрика Оллмана *Sendmail Installation and Operations Guide* (находится в каталоге doc/op дистрибутива; мы иногда ссылаемся на нее как на "руководство по инсталляции") и файл README (находится в каталоге cf).

Использование препроцессора m4

Сначала мы кратко опишем возможности препроцессора m4, а затем покажем, как создать конфигурационный файл из файла макросов. После этого будет рассмотрен ряд важных макросов, которые входят в состав дистрибутива sendmail.

Препроцессор m4 задумывался как внешний интерфейс для языков программирования, который позволил бы разработчикам писать более понятные программы. Это достаточно мощное средство решения многих задач, связанных с преобразованием входной информации. В частности, препроцессор очень хорошо работает с файлами конфигурации sendmail.

Макросы m4 имеют следующий синтаксис:

имя(*аргумент1*, *аргумент2*, ..., *аргументN*)

Между именем и открывающей скобкой не должно быть пробела. Строковые аргументы должны браться в одинарные кавычки. Отметим, что правила употребления кавычек в макросах отличаются от правил, принятых в других языках программирования, так как для левой и правой кавычек используются разные символы[24]. Допускается вложенное употребление кавычек. Удивительно, как препроцессор m4 с таким экзотическим синтаксисом смог "выжить" в борьбе с современными компиляторами.

У препроцессора m4 есть несколько встроенных макросов. Кроме того, пользователи могут определять собственные макросы. Встроенные макросы, которые наиболее часто встречаются в файлах конфигурации программы sendmail, перечислены в табл. 18.6.

Таблица 18.6. Макросы m4, часто применяемые для конфигурирования программы sendmail

Макрос	Назначение
define	Определяет макроконстанту с именем *аргумент1* и значением *аргумент2*
undefine	Отменяет предыдущее определение макроконстанты с именем *аргумент1*
include	Подключает внешний файл с именем *аргумент1*
dnl	Отключает интерпретацию всех символов до следующего символа новой строки включительно
divert	Управляет выходными потоками

[24] Символы кавычек можно поменять с помощью макроса changequote, но это приведет к тому, что макросы m4 нельзя будет применять в программе sendmail, поскольку различные макросы делают скрытые предположения о том, какие именно кавычки используются.

Некоторые администраторы добавляют макрос dnl (от “delete to newline”, т.е. “удалить до символа новой строки”) в конец каждой строки, с тем чтобы транслированный cf-файл выглядел аккуратнее. При отсутствии этого макроса препроцессор m4 добавляет в файл пустые строки. Они не влияют на работу программы sendmail, но делают файл конфигурации трудным для чтения. Мы не включили макросы dnl в наши примеры.

Препроцессор m4 не очень любит комментарии в файлах. К примеру, комментарий

```
# And then define the ...
```

будет воспринят неправильно, так как define является ключевым словом препроцессора. В данном случае следует использовать макрос dnl:

```
dnl # And then define the ...
```

При этом не забудьте вставить пробел между именем макроса и символом комментария.

Файлы для конфигурирования программы sendmail

В дистрибутиве sendmail есть подкаталог cf, в котором находится все необходимое для конфигурирования средствами препроцессора m4, в частности файл README и несколько подкаталогов, перечисленных в табл. 18.7.

Таблица 18.7. Каталоги, содержащие файлы для настройки программы sendmail

Каталог	Содержимое
cf	Примеры mc-файлов (главные конфигурационные файлы)
domain	Примеры доменных файлов для различных доменов в университете Беркли
feature	Файлы, реализующие различные возможности программы
hack	Специальные средства
m4	Базовый файл конфигурации и другие основные файлы
ostype	Файлы для конкретных операционных систем
mailer	Файлы, задающие конфигурацию наиболее распространенных агентов доставки
sh	Сценарии интерпретатора команд, используемые препроцессором m4

Каталог cf/cf содержит примеры mc-файлов, однако их так много, что можно запутаться. Мы рекомендуем хранить свои локальные mc-файлы отдельно от дистрибутива. Создайте новый каталог с именем cf/*название_организации* либо переименуйте каталог cf в cf.examples и создайте новый каталог cf. Во втором случае следует скопировать в новый каталог сценарии Makefile и Build, чтобы инструкции, содержащиеся в файле README, оставались актуальными. Лучше вообще скопировать все конфигурационные mc-файлы в центральное хранилище и не оставлять их в структуре каталогов дистрибутива sendmail. Относительные пути, указанные в сценарии Build, потребуется изменить, если cf-файл будет создаваться на основе mc-файла, не находящегося в иерархии каталогов дистрибутива.

Создание файла конфигурации из готового mc-файла

Прежде чем перейти к описанию всевозможных макросов, средств и опций, давайте взглянем на картину в целом и научимся создавать простейшую конфигурацию без всяких излишеств. Это будет файл конфигурации для одиночного хоста foo.com. Назовем наш файл `foo.mc`.

Указанный файл мы поместим в новый каталог `cf`. Преобразованный (средствами препроцессора `m4`) файл конфигурации будет называться `foo.cf`, находиться в том же каталоге и инсталлироваться под именем `sendmail.cf` в каталог `/etc` либо `/etc/mail`. Последний является стандартным местом нахождения конфигурационного файла, но во многих дистрибутивах для этой цели используется каталог `/etc`.

Существует ряд шаблонов, которые должны присутствовать в каждом новом `mc`-файле:

```
divert(-1)
#### базовый mc-файл для хоста foo.com
divert(0)
VERSIONID(`$Id$')
```

Если в начало файла требуется поместить комментарии, то первая строка должна иметь такой вид:

```
divert(-1)
```

Эта инструкция будет отсеивать всю ненужную информацию в выходном потоке препроцессора `m4`, устраняя потребность в макросе `dnl`. Далее располагаются комментарии (начинаются символом `#`), дополняемые другой инструкцией `divert`.

Стандартная часть файла завершается макросом `VERSIONID` (здесь используется формат RCS). Эта директива подробно рассматривается в следующем разделе.

В большинстве случаев файл конфигурации заканчивается заданием системно-зависимых параметров (макрос `OSTYPE`) и агентов доставки (макрос `MAILER`):

```
OSTYPE(`linux')
define(`confCOPY_ERRORS_TO', `postmaster')
MAILER(`local')
MAILER(`smtp')
```

Кроме того, здесь устанавливается опция `confCOPY_ERRORS_TO`, определяющая, что копии заголовков всех сообщений, которые не удалось доставить, должны быть посланы локальному администратору электронной почты. Это позволит ему вмешиваться в процесс функционирования почты при возникновении проблем локального масштаба.

Для создания файла конфигурации необходимо запустить сценарий `Build`, который мы скопировали в новый каталог `cf`:

```
# ./Build foo.cf
```

либо

```
# make foo.cf
```

Наконец, следует инсталлировать файл `foo.cf` в нужное место. Обычно он превращается в файл `/etc/mail/sendmail.cf`, однако в Red Hat и SUSE он становится файлом `/etc/sendmail.cf`.

В крупной организации можно в каталоге `cf/domain` создать отдельный доменный файл (с расширением `.m4`), содержащий стандартные общесистемные установки.

Пользователи смогут подключать содержимое этого файла к своим конфигурационным файлам. Не каждый компьютер нуждается в отдельном файле конфигурации, однако каждая группа похожих систем (имеющих одинаковую платформу и роль: сервер, клиент и т.п.), вероятно, должна иметь свою конфигурацию.

Изменение конфигурации программы sendmail

Часто бывает, что существующая конфигурация программы `sendmail` правильна, но нужно внести незначительное изменение: попробовать новое средство, добавить новое правило фильтрации спама и т.д. Для этого:

- отредактируйте `mc`-файл, внеся требуемое изменение;
- перестройте файл конфигурации, запустив сценарий `Build` в конфигурационном каталоге;
- инсталлируйте полученный `cf`-файл под именем `sendmail.cf` в нужный каталог;
- пошлите программе `sendmail` сигнал HUP, чтобы заставить ее повторно загрузить свой конфигурационный файл[25].

Даже сейчас, когда процесс конфигурирования программы `sendmail` значительно упростился, решение о том, какой должна быть конфигурация почтовой системы, приходится принимать самостоятельно. Читая о возможностях этой программы, подумайте, как эти возможности "впишутся" в структуру вашей системы. В небольшой организации, скорее всего, будет только хост-концентратор и несколько одиночных хостов, поэтому понадобятся всего две версии файла конфигурации. В более крупной организации, очевидно, потребуются отдельные концентраторы для входящей и исходящей почты и, может быть, сервер POP/IMAP.

Независимо от того, насколько сложной является почтовая система и в каком виде (открытом, за брандмауэром или в виде виртуальной частной сети) она представлена внешнему миру, в каталоге `cf` всегда найдутся готовые фрагменты файла конфигурации, которые после незначительных изменений можно будет использовать.

18.8. Базовые примитивы конфигурации программы sendmail

Конфигурационные команды программы `sendmail` чувствительны к регистру символов. В соответствии с общепринятыми соглашениями имена стандартных макросов пишутся прописными буквами (например, `OSTYPE`), а команды препроцессора `m4` — строчными (например, `define`). Имена конфигурационных переменных начинаются с префикса `conf`, за которым следует описательная часть имени, набранная прописными буквами (например, `confCOPY_ERRORS_TO`).

Макрос обычно ссылается на файл `../`*`имя_макроса/arg1`*`.m4`. К примеру, макрос `OSTYPE(`linux')` вызывает подключение файла `../ostype/linux.m4`.

Для начала рассмотрим ключевые конфигурационные макросы.

---

[25] Воспользуйтесь для этого командой `kill`. Идентификатор процесса `sendmail` хранится в файле `sendmail.pid`. Местонахождение этого файла зависит от дистрибутива (наиболее вероятный вариант — `/var/run/sendmail.pid`). Пример соответствующей команды приводился в конце раздела 18.6..

Макрос VERSIONID

Файлы конфигурации необходимо вести с применением системы контроля изменений CVS или RCS. Это нужно не только для того, чтобы иметь возможность вернуться к одной из предыдущих версий, но и для того, чтобы можно было определить версии файлов макросов, задействованных в формировании файла конфигурации. Для автоматического встраивания информации о версии используется макрос VERSIONID. Синтаксис его вызова таков:

```
VERSIONID(`$Id$')
```

Реальная информация о версии вносится системой RCS на этапе компиляции файла. В итоговом файле sendmail.cf она появляется в виде комментария. Эта информация может оказаться полезной и в ситуации, когда пользователь забыл, куда он поместил дистрибутив sendmail (во многих случаях местонахождение файлов определяется не структурой файловой системы, а наличием свободного дискового пространства).

Макрос OSTYPE

Файлы в каталоге ostype носят имена, соответствующие тем операционным системам, для которых они содержат установки по умолчанию. В частности, в них хранится информация о местонахождении файлов почтовой системы и важных команд, флагах агентов доставки и т.п.

По соглашению эта информация встраивается в конфигурационный файл с помощью макроса OSTYPE[26]. Вызов макроса стоит в начале файла, обычно после макроса VERSIONID.

Файлы каталога ostype в основном содержат определения переменных препроцессора m4. Например, команда

```
define(`ALIAS_FILE', `/etc/aliases')
```

задает местонахождение общесистемного файла псевдонимов. Стандартные установки можно переопределить в mc-файле. Только не изменяйте сам файл, находящийся в каталоге ostype, за исключением случаев, когда он действительно неправилен (при этом нужно послать отчет о найденной ошибке по адресу sendmail-bugs@sendmail.org). В некоторых организациях стремятся унифицировать расположение файла aliases для разных платформ и потому переопределяют его местонахождение в доменном файле.

В файле cf/README перечислены все переменные, которые могут быть определены в файлах каталога ostype. Важнейшие из них приведены в табл. 18.8. В таблицу также включены переменные, связанные с фильтрацией спама (по умолчанию они не определены).

Если программа sendmail инсталлируется в новой версии операционной системы или в новой среде, создайте соответствующий файл в каталоге ostype и отправьте его на хост www.sendmail.org, чтобы файл был включен в последующие версии программы. Моделируйте новый файл на основе существующих и сверяйтесь с таблицей стандартных значений из файла cf/README. Если значение переменной в новой системе совпадает со значением по умолчанию, то нет необходимости повторно определять ее (можно, правда, обезопасить себя на случай изменения стандартных установок).

[26] А где же определен сам макрос OSTYPE? В файле, находящемся в каталоге cf/m4. Этот файл присоединяется к конфигурационному файлу при вызове сценария Build.

Таблица 18.8. Стандартные значения переменных, устанавливаемых в файлах каталога ostype

Переменная	Значение по умолчанию
ALIAS_FILE	/etc/mail/aliases
HELP_FILE	/etc/mail/helpfile
STATUS_FILE	/etc/mail/statistics
QUEUE_DIR	/var/spool/mqueue
MSP_QUEUE_DIR	/var/spool/clientmqueue
LOCAL_MAILER_PATH	/bin/mail
LOCAL_SHELL_PATH	/bin/sh
LOCAL_MAILER_MAX	не определена
LOCAL_MAILER_MAXMSGS	не определена
SMTP_MAILER_MAX	не определена
SMTP_MAILER_MAXMSGS	не определена

Файлы каталога ostype для наших демонстрационных дистрибутивов перечислены в табл. 18.9.

Таблица 18.9. Файлы каталога ostype для Linux

Система	Файл	Каталог	Строка вызова
Red Hat	linux.m4	/usr/share/sendmail-cf	OSTYPE(`linux')
Fedora	linux.m4	/usr/share/sendmail-cf	OSTYPE(`linux')
SUSE	suse-linux.m4	/usr/share/sendmail	OSTYPE(`suse-linux')
Debian	debian.m4	/usr/share/sendmail/sendmail-cf	OSTYPE(`debian')
Ubuntu	linux.m4	/usr/share/sendmail	OSTYPE(`linux')

В SUSE дистрибутив sendmail находится в каталоге /usr/share/sendmail. Файл suse-linux.m4 помещен в подкаталог ostype и не является частью дистрибутива, доступного на хосте www.sendmail.org. Это очень большой файл (более 80 строк), он содержит объявления многочисленных средств и макроконстант, более привычных для главного конфигурационного файла организации. В результате реальная конфигурация оказывается скрытой от системного администратора. Мы не рекомендуем применять такой подход.

В Debian конфигурационные файлы скрыты в каталоге /usr/share/sendmail/sendmail.cf/. Каталог sendmail.cf (странный выбор имени, не правда ли?) соответствует каталогу cf дистрибутива sendmail и содержит все необходимое, включая файл ostype/debian.m4. Этот файл состоит из 50 строк, которые задают имена файлов либо являются комментариями. Многие установки идентичны стандартным установкам дистрибутива sendmail, т.е. являются избыточными. В то же время это защищает Debian от изменения стандартных параметров программы, что могло бы привести к возникновению несогласованностей и ошибок. Единственный серьезный недостаток файла ostype/debian.m4 заключается в том, что к нему подключается общий доменный файл, входящий в дистрибутив sendmail. Подобное здесь совершенно неуместно: вызов макроса DOMAIN должен стоять в начале mc-файла, а не быть спрятанным в файле каталога ostype.

Макрос DOMAIN

Макрос DOMAIN позволяет хранить общесистемную информацию в одном файле (cf/domain/*имя_файла*.m4) и ссылаться на него в файлах конфигурации каждого компьютера следующим образом:

```
DOMAIN(`имя_файла')
```

Имя файла обычно выбирается так, чтобы в нем отражалось название организации. Например, наш файл для факультета вычислительной техники (Computer Science department) называется cs.m4:

```
DOMAIN(`cs')
```

Как и OSTYPE, макрос DOMAIN является хорошим средством подключения внешних файлов. Он делает структуру конфигурационного файла понятнее и позволяет в будущем легко модифицировать конфигурацию. Макрос особенно полезен, когда все cf-файлы централизованно создаются из mc-файлов, хранящихся в одном месте.

В небольших организациях необходимости в доменном файле, как правило, нет, а вот крупные организации часто используют такие файлы для указания хостов-ретрансляторов и установки общесистемных опций защиты, а также для определения таблиц агентов доставки, виртуальных доменов и пр.

Доменный файл, включенный в дистрибутив sendmail, содержит примеры записей, которые обычно помещают в общесистемные файлы. Вот его текст (комментарии и макросы dnl удалены):

```
VERSIONID(`$Id: generic.m4,v 8.15 1999/04/04 00:51:09 ca Exp $')
define(`confFORWARD_PATH', `$z/.forward.$w+$h:$z/.forward+$h:
       $z/.forward.$w:$z/.forward')
define(`confMAX_HEADERS_LENGTH', `32768')
FEATURE(`redirect')
FEATURE(`use_cw_file')
EXPOSED_USER(`root')
```

Здесь задается путь к пользовательским файлам переадресации, ограничивается длина заголовков[27], а также включаются средства redirect для обслуживания пользователей, недавно покинувших организацию, и use_cw_file для обработки эквивалентных имен компьютеров. Если в mc-файле включен режим маскирования адресов, то благодаря макросу EXPOSED_USER это не коснется пользователя root. Каждая из соответствующих конструкций будет подробно описана далее.

Макрос MAILER

Макрос MAILER необходимо включать для каждого агента доставки, который требуется активизировать. Полный перечень поддерживаемых агентов находится в каталоге cf/mailers дистрибутива sendmail. В настоящее время поддерживаются следующие агенты: local, smtp, fax, usenet, procmail, qpage, cyrus, pop, phquery и uucp.

[27] Хакеры применяли сверхдлинные заголовки для организации атак типа “отказ в обслуживании” в старых версиях программы sendmail. Соответствующая строка оставлена здесь на тот случай, если кто-то из читателей до сих пор использует небезопасную версию (более раннюю, чем 8.9.3).

Обычно минимальный набор таков:

```
MAILER(`local')
MAILER(`smtp')
```

Первая директива активизирует агенты `local` и `prog`, а вторая — агенты `smtp`, `esmtp`, `dsmtp`, `smtp8` и `relay`. Адреса вида *пользователь+уточнение@организация.домен* стали поддерживаться локальным агентом начиная с версии 8.7[28]. Компонент *пользователь* определяет почтовый ящик, куда следует доставлять сообщения, а компонент *уточнение* представляет собой дополнительный параметр, на основании которого локальный агент, к примеру программа `procmail`, может сортировать входящую почту.

Планируя менять значение макроконстанты, связанной с агентами доставки (скажем, `USENET_MAILER_ARGS` или `FAX_MAILER_PATH`), убедитесь в том, что строка, в которой устанавливается макроконстанта, *предшествует* строке вызова самого агента. В противном случае будет использовано старое значение. По этой причине макросы `MAILER` размещают ближе к концу файла конфигурации.

Агент `pop` взаимодействует с программой `spop`, которая является частью пакета `mh` и реализует протокол POP, определенный в документе RFC1460. Агент `pop` применяется персональными компьютерами и системами Macintosh для доступа к почте, хранящейся в UNIX-системе. Агент `cyrus` используется сервером IMAP разработки университета Карнеги-Меллона и бывает двух видов: собственно `cyrus` (доставляет сообщения в пользовательские почтовые ящики) и `cyrusbb` (доставляет сообщения в центральную доску объявлений). Этот агент тоже понимает синтаксис *пользователь+уточнение* и должен активизироваться после агента `local`.

Агент `fax` интегрирует пакет HylaFAX в систему электронной почты. В SUSE этот пакет хранится в каталоге `/usr/bin/faxmail`; в Red Hat и Debian он отсутствует. При отправке почты по адресу *пользователь@получатель*.fax тело сообщения передается как факс-документ. В качестве аргумента *получатель*, как правило, задается номер телефона. Чтобы разрешить указание символических имен вместо телефонных номеров, используйте индексированную базу данных.

🕮 Пакет HylaFAX можно найти на Web-сайте `www.hylafax.org`.

Интегрировать программы HylaFAX и `sendmail` следует путем инсталляции специального сценария из дистрибутива HylaFAX в каталог `/usr/local/bin`. Если требуется, измените значение макроконстанты `FAX_MAILER_PATH`. Для передачи входящих факсов из буферной области в почтовый ящик пользователя в любом случае необходимо вмешательство человека. Факс-документы можно преобразовывать в формат PostScript (с помощью программы HylaFAX) и просматривать с помощью GNU-программы `ghostscript`.

🕮 Программа `ghostscript` доступна на Web-сайте `www.gnu.org`.

Агент `qpage` (`www.qpage.org`) взаимодействует с программой QuickPage для доставки почты на пейджер.

Описанные выше макросы `VERSIONID`, `OSTYPE` и `MAILER` — это все, что необходимо для создания базового файла *имя_компьютера*`.mc`.

[28] Синтаксис *пользователь+уточнение* был придуман в университете Карнеги-Меллона, где он применялся локальными утилитами для маршрутизации и сортировки почты.

18.9. Дополнительные примитивы конфигурации программы sendmail

В этом разделе мы продолжим описание макросов и, кроме того, рассмотрим некоторые средства, используемые для изменения стандартного поведения программы sendmail. Будет также показано, как на уровне конфигурационных настроек решать ряд административных задач, в частности маскировать адреса, создавать виртуальные домены, управлять безопасностью и бороться со спамом.

Макрос FEATURE

Макрос FEATURE позволяет активизировать некоторые распространенные опции (называемые *средствами*) путем подключения m4-файлов из каталога feature. Мы поместили описание средств макроса FEATURE и ряда других макросов программы sendmail вместе, поскольку они тесно связаны друг с другом. Даже когда возможность конфигурирования программы sendmail при помощи препроцессора m4 была нова, описание макроса FEATURE занимало значительную часть главы об электронной почте. На сегодняшний день список средств макроса FEATURE столь велик, что для их полного рассмотрения потребовалась бы целая глава. Синтаксис макроса таков:

```
FEATURE(ключевое_слово, арг, арг, ...)
```

где *ключевое_слово* обозначает файл *ключевое_слово*.m4 в каталоге cf/feature, а *арг* — это аргументы, записываемые в файл. Полный перечень средств макроса FEATURE можно найти в файле cf/README. Ниже описаны наиболее часто используемые средства.

Средство use_cw_file

Внутренний класс w программы sendmail содержит имена всех локальных хостов, для которых компьютер принимает и доставляет почту. Клиентский компьютер может включить в этот класс свое имя, псевдонимы и адрес интерфейса localhost. Если же конфигурируемый компьютер является почтовым концентратором, то его класс w должен также включать имена всех локальных хостов и виртуальных доменов, для которых он принимает почту.

Средство use_cw_file определяет класс w на основании содержимого файла /etc/mail/local-host-names (который раньше назывался sendmail.cw; точное имя файла задается опцией confCW_FILE). Без этого средства программа sendmail принимает только почту, адресованную компьютеру, на котором она работает. На сервере входящей почты файл local-host-names должен содержать список всех компьютеров и доменов, для которых сервер обрабатывает почту. После внесения изменений в этот файл необходимо послать программе sendmail сигнал HUP, так как она читает файл лишь в процессе запуска.

Команда

```
FEATURE(`use_cw_file')
```

активизирует описываемое средство и подключает файл local-host-names. Вот его пример:

```
# local-host-names -- укажите здесь все псевдонимы компьютера.
toadranch.com
```

```
coyote.toadranch.com
big-tr.com
yoherb.com
herbmorreale.com
appliedtrust.com
applied-trust.com
atrust.com
```

Все перечисленные здесь адреса — это виртуальные домены, обслуживаемые локальным компьютером.

Средство redirect

Если кто-то из сотрудников покинул организацию, его почту нужно либо пересылать по новому адресу, либо возвращать отправителю с сообщением об ошибке. Средство `redirect` реализует более элегантный способ возврата почты. Пусть, к примеру, некто Джо Смит (Joe Smith) закончил учебу в университете (oldsite.edu) и поступил на работу в компанию (newsite.com). Тогда включение средства `redirect` с помощью команды

```
FEATURE(`redirect')
```

и добавление строки

```
smithj: joe@newsite.com.REDIRECT
```

в файл `aliases` обеспечат возврат почты, приходящей на имя пользователя `smithj`, отправителю вместе с сообщением об ошибке, в котором предлагается посылать почту по адресу `joe@newsite.com`. Сами почтовые сообщения пересылаться не будут.

Средство always_add_domain

Когда средство `always_add_domain` включено, программа `sendmail` добавляет имя локального хоста к адресу локального получателя, если адрес не является полностью определенным. Предположим, что пользователь lynda (`lynda@cs.colorado.edu`) посылает сообщение локальным пользователям barb и evi. Если средство `always_add_domain` не включено, в заголовках сообщения адреса отправителя и получателя будут представлены в виде регистрационных имен. При включенном средстве `always_add_domain` адреса будут полностью определены еще до того, как сообщение покинет компьютер пользователя lynda.

Средство эффективно в том случае, когда буферные каталоги совместно используются компьютерами, не имеющими единого файла псевдонимов или не содержащими одинаковые файлы `passwd` (в этой ситуации, очевидно, не следовало бы совместно использовать почтовые буферы). Поскольку адресат (псевдоним или пользователь) будет известен не на каждом компьютере, то отправитель должен полностью определить адрес, чтобы по нему можно было послать ответное сообщение.

Еще один аргумент в пользу средства `always_add_domain` состоит в том, что сообщения с неполными именами часто отбрасываются как спам. Поэтому мы рекомендуем всегда активизировать это средство (если, конечно, вы не спамер!).

Когда используется режим маскирования адресов (макрос `MASQUERADE_AS`), включение средства `always_add_domain` приведет к тому, что добавляться будет имя хоста, выбранного для маскирования, а не имя локального хоста. Это может вызвать проблемы, если файл `aliases` или `passwd` на локальном компьютере не является подмножеством одноименного файла на хосте, который указан в качестве маскирующего.

Средство nocanonify

Программа sendmail обычно проверяет, является ли доменная часть адреса полностью определенным именем или это запись CNAME. Во втором случае осуществляется раскрытие адреса. Происходит это путем поиска почтового имени в DNS. Средство nocanonify отменяет такой поиск. Например, если в организации используется главный концентратор почты, через который проходит вся локальная почта, то на клиентских компьютерах можно задать инструкцию

```
FEATURE(`nocanonify')
```

и тем самым отменить поиск в DNS на локальном уровне. Программа sendmail не следит за тем, ведется ли поиск в DNS по мере продвижения сообщения от компьютера к компьютеру в пределах одной организации, — у нее просто нет такой возможности. Таким образом, средство nocanonify позволяет согласовать все операции поиска. Этот прием иллюстрируется во втором примере (см. раздел 18.11).

Средство nocanonify может также использоваться на очень больших почтовых хостах, где функционируют агенты подачи и транспортировки почты. В этом случае поиск в DNS осуществляет агент подачи, а главный компьютер, на котором запущен агент транспортировки, работает в режиме nocanonify.

Иногда поиск в DNS нежелателен по причине затрат ресурсов, но в то же время допустим поиск в локальном домене. Чтобы исключить домены из спецификации nocanonify, воспользуйтесь макросами CANONIFY_DOMAIN либо CANONIFY_DOMAIN_FILE, которые в качестве аргумента принимают, соответственно, список доменов или имя файла. Например, строки

```
FEATURE(`nocanonify')
CANONIFY_DOMAIN(`cs.colorado.edu cs')
```

запрещают поиск в DNS, за исключением адресов вида *пользователь*@cs.colorado.edu и *пользователь*@cs. Эти макросы впервые появились в версии 8.12.

Таблицы и базы данных

У программы sendmail есть несколько средств, использующих конструкцию, которая условно называется таблицей. В основном таблица представляет собой текстовый файл с информацией о маршрутах, псевдонимах и методах доступа, который преобразуется при помощи команды makemap в формат базы данных и затем применяется программой sendmail в качестве внутренней поисковой базы данных. Не обязательно, чтобы источником данных служил файл; данные для таблиц программы sendmail могут поступать от DNS, NIS, LDAP и других служб. В то же время наличие централизованного IMAP- или POP-сервера избавляет программу от рутинных операций по поиску пользователей и делает ненужными некоторые из описанных ниже таблиц. Доступные таблицы будут перечислены далее в табл. 18.10.

Поддерживаются две библиотеки функций для работы с базами данных: dbm/ndbm (является стандартом в большинстве версий Linux) и Berkeley DB (расширяемая библиотека, поддерживающая несколько схем хранения). Выбор библиотеки необходимо сделать на этапе компиляции. Мы рекомендуем библиотеку Berkeley DB (доступна на Web-сайте www.sleepycat.com), поскольку эта СУБД работает быстрее и создает меньшие по объему файлы.

Имеются три типа поисковых таблиц:

- dbm — используется расширяемый алгоритм хеширования (dbm/ndbm);
- hash — используется стандартная схема хеширования (Berkeley DB);
- btree — используется структура данных типа бинарного дерева (Berkeley DB).

Для большинства задач, связанных с применением таблиц, наиболее приемлемым оказывается тип hash, который предлагается по умолчанию. Используйте команду makemap для создания базы данных из текстового файла. При этом нужно указать тип базы данных и имя ее файла. Текстовая версия базы данных должна подаваться на стандартный вход команды makemap, например:

```
# makemap hash /etc/mail/access < /etc/mail/access
```

На первый взгляд, команда выглядит неправильной, так как кажется, будто в результате произойдет замена входного файла пустым выходным. Однако команда makemap присоединяет к имени выходного файла расширение, т.е. в действительности он называется /etc/mail/access.db — и никакого конфликта нет. При каждом изменении исходного текстового файла необходимо с помощью команды makemap перестраивать файл базы данных (посылать программе sendmail сигнал HUP не нужно).

В большинстве случаев при поиске ключа в базе данных выбирается самое длинное из возможных совпадений, поэтому порядок записей текстового файла не имеет значения, что свойственно хешированным структурам данных. Если какое-то средство ожидает файл базы данных в качестве параметра, то по умолчанию принимается тип hash и файл /etc/mail/*имя_таблицы*.db. Чтобы изменить эту установку, укажите требуемый тип при вызове команды makemap и в макросе FEATURE либо переопределите переменную DATABASE_MAP_TYPE:

```
define(`DATABASE_MAP_TYPE', `dbm')
```

Для подключения новой базы данных access.db необходимо добавить в mc-файл следующую команду:

```
FEATURE(`access_db', `hash /etc/mail/access')
```

Поскольку в этой строке активизируются установки по умолчанию, ее можно упростить:

```
FEATURE(`access_db')
```

Имя файла базы данных разрешается указывать как с расширением (.db), так и без него.

Не забывайте перестраивать базу данных при помощи команды makemap всякий раз, когда изменяется исходный текстовый файл. В противном случае изменения не будут учтены.

Ниже будут описаны средства mailertable, generictable и virtusertable. Средство access_db описано в разделе, посвященном борьбе со спамом. Средство user_db не рассматривается вовсе, так как оно устарело и скоро будет удалено.

Начиная с версии 8.12 все поисковые таблицы и классы могут иметь в качестве источника базу данных LDAP. Это означает, что можно попросить программу sendmail связаться с сервером LDAP для определения правил маршрутизации почты и перезаписи заголовков. Достаточно лишь указать LDAP в качестве аргумента средства access_db:

```
FEATURE(`access_db', `LDAP')
```

Эта строка заставит программу использовать стандартную схему LDAP, содержащуюся в файле `cf/sendmail.schema`. Разрешается также определять собственные схемы баз данных (подробнее об этом читайте в файле `cf/README`).

Средство mailertable

Средство `mailertable` организует перенаправление почты, адресованной конкретному хосту или домену, по альтернативному адресу с помощью заданного агента доставки. Оно применяется для отправки почты и принимает во внимание только доменную часть адреса, не учитывая имя пользователя. Адрес заголовка при этом не переписывается, поэтому почта, предназначенная тому же пользователю, посылается на другой хост другим агентом доставки. Первоначально средство `mailertable` предназначалось для взаимодействия с иными почтовыми системами, такими как UUCP, DECnet и BITNET, но сегодня оно используется в основном для перенаправления почты от шлюза на внутренний сервер или на сервер, расположенный в другой организации, у которой нет прямого доступа к сети Internet.

Чтобы включить средство `mailertable`, вставьте в `mc`-файл следующую строку:

```
FEATURE(`mailertable')
```

Запись в таблице агентов доставки имеет следующий вид:

```
старый_домен    агент_доставки:получатель
```

Если перед левым ключевым словом указана точка, то это означает "любой хост домена". Ключами могут служить только имена компьютеров или доменов; имена пользователей недопустимы. Поле *получатель* может содержать доменное имя, выражение *пользователь@домен* или вообще быть пустым. В последнем случае адреса конверта не меняются. В поле *агент_доставки* должно быть указано имя агента доставки, определенное с помощью макроса `MAILER` (см. конец раздела 18.8).

Предположим, что основным внутренним почтовым сервером организации является программа MS Exchange, которая не должна иметь выход в Internet. Тогда почтовым шлюзом можно сделать компьютер, работающий под управлением Linux. С этого шлюза на сервер MS Exchange должна пересылаться вся входящая почта (после проверки на вирусы и другой предварительной обработки). Соответствующая запись в таблице агентов доставки будет такой (подразумевается, что сервер Exchange имеет показанный ниже внутренний IP-адрес):

```
my-domain       esmtp:[192.168.1.245]
```

С другой стороны, это ведь один из способов ретрансляции, которая, как будет показано в разделе 18.10, должна находиться под неусыпным контролем. Таким образом, чтобы завершить пример, следует добавить запись

```
To: my-domain  RELAY
```

в базу доступа, разрешив тем самым ретрансляцию всей почты, адресованной пользователям домена `my-domain`.

Средство genericstable

Применение средства `genericstable` аналогично созданию псевдонимов для исходящей почты. Например, можно заменить имя `trent@xor.com` в отправляемом сооб-

щении именем trent.hein@xor.com. При этом переписываются адреса заголовка, а не конверта, т.е. средство не влияет на то, куда будет доставлена почта. Оно определяет, кто получит ответ.

Существует несколько способов подобной подстановки, но только средство genericstable позволяет помещать в ключ замены как имя пользователя, так и имя компьютера. К адресам, заданным в средстве genericstable, могут также применяться описанные ниже средства masquerade_envelope и allmasquerade.

Перед использованием средства genericstable убедитесь в том, что нужный домен относится к классу generics. Чтобы включить домен в этот класс, воспользуйтесь макросом GENERICS_DOMAIN или укажите имя домена в файле, имя которого задано в макросе GENERICS_DOMAIN_FILE.

Следующие команды в mc-файле активизируют средство genericstable со стандартными установками:

```
GENERICS_DOMAIN_FILE(`/etc/mail/local-host-names')
FEATURE(`genericstable')
```

В данном примере в класс generics включается имя любого компьютера, для которого принимается почта. Средство genericstable замедляет работу программы sendmail, поскольку требует просматривать каждый адрес отправителя.

Средство virtusertable

Таблица виртуальных пользователей содержит доменные псевдонимы для входящей почты. Средство virtusertable позволяет на одном компьютере работать с несколькими виртуальными доменами. Оно часто применяется в организациях, занимающихся Web-хостингом.

Поле ключа таблицы содержит адрес электронной почты (*пользователь@компьютер.домен*) или спецификацию домена (*@домен*), а поле значения — локальный или внешний почтовый адрес. Если ключ — это имя домена, то имя пользователя в поле значения может задаваться в виде переменной %1 либо указываться явно. Если же пользовательская часть спецификации ключа имеет вид *пользователь+уточнение*, то переменная %2 будет содержать фрагмент *уточнение*, а переменная %3 — фрагмент *+уточнение*.

Рассмотрим следующие примеры:

```
info@foo.com   foo-info            # маршрут к локальному пользователю
info@bar.com   bar-info            # другой локальный пользователь
joe@bar.com    error:No such user  # возврат ошибки
@baz.org       jane@elsewhere.com  # вся почта направляется пользователю jane
@zokni.org     %1@elsewhere.com    # вся почта направляется тому же
                                   # пользователю в другом домене
```

Все хосты, перечисленные в левой части, должны быть указаны в файле /etc/mail/local-host-names (или в классе VirtHost). В противном случае программа sendmail попытается найти хост в Internet и доставить почту туда. Если DNS направит программу sendmail обратно на сервер, будет получено сообщение об ошибке локальной конфигурации. К сожалению, программа sendmail не может сказать, что в действительности сообщение об ошибке должно выглядеть так: “В файле класса w отсутствует ключ для средства virtusertable”.

Чтобы описанный механизм работал, необходимо соблюдение следующих условий.

- Должны существовать такие записи MX базе данных DNS, чтобы почта доставлялась нужному хосту.
- В файле `/etc/mail/local-host-names` должны быть такие записи, чтобы локальный компьютер мог принять почту. С этой же целью может быть задан макрос `VIRTUSER_DOMAIN` (или эквивалентный ему макрос `VIRTUSER_DOMAIN_FILE`).
- Таблица виртуальных пользователей должна указывать программе `sendmail`, что делать с почтой.

Средство `virtusertable` активизируется так:

```
FEATURE(`virtusertable')
```

В разделе 18.11 содержатся примеры, демонстрирующие использование средства `virtusertable` для организации виртуального хостинга.

Средство ldap_routing

Рассказывая о псевдонимах и перенаправлении почты, нельзя не упомянуть еще раз протокол LDAP. Этот протокол может быть использован вместо средства `virtusertable` для маршрутизации и приема почты в виртуальных доменах. Он также позволяет управлять псевдонимами, поисковыми таблицами и классами, а в версии 8.12 — еще и списками рассылки.

Чтобы использовать протокол LDAP для решения указанных задач, необходимо добавить в конфигурационный файл несколько команд, а затем скомпилировать программу `sendmail` с поддержкой протокола. В `mc`-файл требуется включить следующие строки:

```
define(`confLDAP_DEFAULT_SPEC', `-h сервер -b базовое_имя')
FEATURE(`ldap_routing')
LDAPROUTE_DOMAIN(`домен')
```

Благодаря им программа `sendmail` узнает о том, что нужно использовать базу данных LDAP для маршрутизации входящей почты, адресованной указанному домену. Опция `confLDAP_DEFAULT_SPEC` определяет сервер LDAP, а также имя, с которого начинается поиск в базе данных.

В следующем примере базовое имя задается так: `o=sendmail.com, c=US`. Если сервер LDAP работает не со стандартным портом 389, добавьте выражение `-p порт` в спецификацию опции `LDAP_DEFAULT_SPEC`.

Программа `sendmail` использует значения двух информационных дескрипторов LDAP:

- `mailLocalAddress` — для получателей входящей почты;
- `mailRoutingAddress` — для псевдонимов адресатов.

Поддерживается также дескриптор `mailHost`. При его наличии почта для хоста, указанного в дескрипторе `mailRoutingAddress`, посылается компьютеру, на который ссылается наиболее приоритетная запись MX хоста, указанного в дескрипторе `mailHost`.

Например, присутствие LDAP-записи (для сервера, сконфигурированного с корневым отличительным именем `cn=root, o=sendmail.com, c=US`) следующего вида:

```
dn: uid=eric, o=senmail.com, c=US
objectClass: inetLocalMailRecipient
mailLocalAddress: eric@sendmail.org
mailRoutingAddress: eric@eng.sendmail.com
```

приведет к тому, что почта, адресованная пользователю eric@sendmail.org (записи MX предписывают посылать ее на хост sendmail.com), будет доставляться по адресу eric@eng.sendmail.com. Если же LDAP-запись будет включать еще и строку

```
mailHost: mailserver.sendmail.com
```

то почта для пользователя eric@sendmail.org будет переадресовываться пользователю eric@eng.sendmail.com и направляться хосту, на который ссылается наиболее приоритетная запись MX компьютера mailserver.

Формат базы данных LDAP допускает использование неполных записей типа @*домен*, которые перенаправляют почту в указанный домен независимо от того, кому конкретно она адресована (как в случае средства virtusertable).

Начиная с версии 8.12 в программе sendmail появился дополнительный конфигурационный примитив — макрос LDAPROUTE_EQUIVALENT (или LDAPROUTE_EQUIVALENT_FILE), позволяющий задавать эквивалентные версии доменного имени, для которого организуется перенаправление почты средствами LDAP. К примеру, при поступлении почты, адресованной пользователю user@host1.mydomain, будет сделан запрос в LDAP относительно адреса user@host1.mydomain, а затем — относительно адреса @host1.mydomain. Если же включить в конфигурационный файл строку

```
LDAPROUTE_EQUIVALENT(`host1.mydomain')
```

то будут также проверены ключи user@mydomain и @mydomain. Эта возможность позволяет использовать единую базу данных для маршрутизации почты в крупной организации.

У средства ldap_routing теперь появились дополнительные аргументы, с помощью которых можно уточнить сведения об используемой схеме LDAP и задать правила обработки пользовательских имен, имеющих компонент +*уточнение*. В версии 8.13 программы sendmail есть новый аргумент sendertoo, благодаря которому можно отклонять почтовые сообщения SMTP, поступающие от отправителя, который не зарегистрирован в LDAP. Как всегда, за подробностями отошлем вас к файлу cf/README.

Маскирование адресов и макрос MASQUERADE_AS

Макрос MASQUERADE_AS позволяет задать единый адрес, за которым будут "скрываться" все компьютеры. Это выглядит так, будто вся почта поступает с одного компьютера или из одного домена. Адрес отправителя *пользователь*@*исходный_хост.домен* меняется на *пользователь*@*маскированное_имя*. Естественно, адреса-маски должны быть легитимными, чтобы можно было получить ответ на свою почту.

Такая конфигурация позволяет всем пользователям организации иметь общий электронный адрес. Если, например, все хосты домена atrust.com маскируются под именем artust.com, то почта, сгенерированная по адресу *пользователь*@*компьютер*.artust.com, будет содержать адрес отправителя *пользователь*@artust.com. Компьютер, который управляет доменом artust.com, должен знать, как доставлять почту всем пользователям, даже тем, у кого нет собственного регистрационного имени на сервере входящей почты. Кроме того, регистрационные имена должны быть уникальными во всем домене.

Некоторые пользователи и адреса (root, postmaster, hostmaster, trouble, operations, Mailer-Daemon и др.) не должны подвергаться маскированию. Это можно сделать с помощью макроса EXPOSED_USER.

Например, последовательность команд

```
MASQUERADE_AS(`artust.com')
EXPOSED_USER(`root')
EXPOSED_USER(`Mail-Daemon')
```

идентифицирует всю почту как поступающую от имени *пользователь*@artust.com, если только она не послана пользователем root или самой почтовой системой (такая почта снабжается реальным адресом отправителя).

В версии 8.12 появилась возможность отменять маскирование для почты, направляемой в локальный домен (или конкретным хостам). Это может быть удобно в организации, где используется незарегистрированное частное доменное имя, поэтому маскировать следует только адреса сообщений, направляемых в Internet. Синтаксис соответствующих макросов таков:

```
FEATURE('local_no_masquerade')
MASQUERADE_EXCEPTION(`хост.домен')
MASQUERADE_EXCEPTION_FILE(`имя_файла')
```

Существует несколько расширений базового макроса MASQUERADE_AS. Они реализованы как в виде макросов, так и в виде средств:

- макрос MASQUERADE_DOMAIN,
- макрос MASQUERADE_DOMAIN_FILE,
- макрос MASQUERADE_EXCEPTION,
- макрос MASQUERADE_EXCEPTION_FILE,
- средство limited_masquerade,
- средство allmasquerade,
- средство masquerade_envelope,
- средство masquerade_entire_domain.

Мы рекомендуем использовать макрос MASQUERADE_AS наряду со средствами allmasquerade и masquerade_envelope. Средство limited_masquerade модифицирует работу макроса MASQUERADE_DOMAIN и применяется в системах виртуального хостинга. Макрос MASQUERADE_DOMAIN позволяет перечислить домены, подлежащие маскированию. Этот список предварительно загружается из класса w, который обычно определяется с помощью средства use_cw_file и содержит перечень хостов домена. Если же включено средство limited_masquerade, то список не будет инициализирован содержимым класса w. Все эти домены будут скрыты за доменным именем-маской.

Средство allmasquerade распространяет режим маскирования на получателей сообщения (а не только на отправителей), а средство masquerade_envelope задает маскирование адресов конверта в дополнение к адресам заголовка[29]. Этих двух средств достаточно для того, чтобы все адреса обрабатывались одинаково. Средство masquerade_entire_domain включает маскирование всех адресов в указанных доменах.

[29] Адреса заголовка — это “To”, “From”, “Cc” и “Bcc”. Адреса конверта определяют, куда в действительности будет доставлено сообщение. Адреса конверта формируются пользовательским агентом из адресов заголовка, однако программа sendmail обрабатывает их отдельно. Многие средства программы, относящиеся к маскированию и перенаправлению почты, нельзя было бы реализовать без учета различий между адресами заголовка и конверта.

Макроконстанты MAIL_HUB и SMART_HOST

Используя средства маскирования, можно сделать так, что почта будет казаться отправленной из одного компьютера или домена. Это достигается путем замены адресов заголовка и в некоторых случаях конверта. Однако иногда требуется, чтобы отправкой (или приемом) почты действительно занимался один компьютер. Для входящей почты это реализуется при помощи макроконстанты MAIL_HUB, а для исходящей — при помощи макроконстанты SMART_HOST.

Если необходимо пересылать всю входящую почту на центральный сервер для последующей ее доставки, присвойте макроконстанте MAIL_HUB значение *агент_доставки*:*компьютер*, где *агент_доставки* — это агент, используемый для передачи почты на указанный компьютер. В отсутствие первого аргумента используется агент relay. Приведем пример:

```
define(`MAIL_HUB', `smtp:mailhub.cs.colorado.edu')
```

Макроконстанта SMART_HOST заставляет компьютер доставлять только локальную почту, а внешнюю — отправлять по указанному адресу. Этой возможностью пользуются компьютеры, которые скрыты за брандмауэром и не могут непосредственно посылать запросы серверу DNS. Синтаксис определения этой константы такой же, как у константы MAIL_HUB. По умолчанию установлен агент доставки relay. Приведем пример:

```
define(`SMART_HOST', `smtp:mailhub.cs.colorado.edu')
```

В этих примерах один и тот же компьютер действует как сервер и входящей, и исходящей почты. В крупных организациях может потребоваться выделить для этих целей разные компьютеры. Компьютер, работающий в режиме SMART_HOST, должен разрешать ретрансляцию, чтобы клиенты могли пересылать через него почту. Записи таблицы mailertable перекрывают установку SMART_HOST.

Маскирование и маршрутизация

Мы уже описали много средств и макросов, которые тем или иным способом воздействуют на адреса электронной почты. Сейчас самое время провести их сравнение. В частности, следует рассмотреть, что именно они делают: изменяют заголовки, адреса конверта или адрес доставки сообщения, применяются к входящей или исходящей почте, к адресам получателя или отправителя и т.д.

Конечно, если бы страница книги была раза в два шире, можно было бы проиллюстрировать все различия. А поскольку это не так, приведем лишь общую информацию (табл. 18.10). За более полными сведениями читателям придется обратиться к документации по программе sendmail.

Элементы, указанные в табл. 18.10 прописными буквами, являются макросами препроцессора m4. Строчными буквами записаны средства, активизируемые посредством макроса FEATURE. Отступами показана зависимость средств и макросов. Например, средства, связанные с макросом MASQUERADE_AS, не работают, пока этот макрос не включен. Средства маскирования воздействуют на адреса в заголовках исходящих сообщений и определяют, можно ли будет получить на них ответ. Средства маршрутизации влияют на доставку сообщений.

Таблица 18.10. Сравнение средств маскирования и маршрутизации

	Конструкция	Почта	Влияние*	Часть адреса
Маскирование	MASQUERADE_AS	исх.	ОЗ	*хост.домен*
	allmasquerade	исх.	ПЗ**	*хост.домен*
	MASQUERADE_DOMAIN[_FILE]	исх.	ОЗ	*хост.домен*
	masquerade_entire_domain	исх.	ОЗ	*хост.поддомен.домен*
	limited_masquerade	исх.	ОЗ	*хост.домен*
	masquerade_envelope	исх.	ОК***	*хост.домен*
	genericstable	исх.	ОЗ	*пользователь@хост.домен*
Маршрутизация	mailertable	исх.	АгАД	*хост.домен*
	virtusertable	вход.	ПД	*пользователь@хост.домен*
	ldap	вход.	ПД	*пользователь@хост.домен*
	mailhub	вход.	ПД	*хост.домен*
	smarthost	исх.	ПД	*хост.домен*

* О — отправитель, П — получатель, Д — доставка, З — заголовок, К — конверт, Аг — агент, А — адрес.

** Если средство allmasquerade включено, то все остальные маскирующие конструкции будут переписывать не только адрес отправителя, но и адрес получателя.

*** Если средство masquerade_envelope включено, то все остальные маскирующие конструкции будут переписывать не только адреса заголовка, но и адреса конверта.

Средство nullclient

Средство nullclient предназначено для компьютера, который не получает почту непосредственно, а всю исходящую почту направляет на центральный сервер. У такого компьютера mc-файл содержит всего две строки:

```
OSTYPE(`тип_ОС')
FEATURE(`nullclient', `почтовый_сервер')
```

Средство nullclient отменяет многие установки, сделанные при помощи других средств. Вся почта без исключения будет доставляться на компьютер *почтовый_сервер* для последующей обработки[30]. Учтите, что сервер должен обеспечивать дальнейшую ретрансляцию почты, поступающей от клиента, если пользователи регулярно отправляют почту с клиентского компьютера и не взаимодействуют с отдельным сервером исходящей почты. В последних версиях программы sendmail ретрансляция отключена по умолчанию. Об управлении ретрансляцией будет подробнее рассказываться в разделе, посвященном борьбе со спамом (раздел 18.10). И еще одно важное замечание. Средство nullclient вызывает маскирование адреса компьютера адресом почтового сервера, поэтому, возможно, понадобится ввести конструкцию EXPOSED_USER для пользователя root.

[30] Если сконфигурировать клиентский компьютер таким образом, а затем протестировать конфигурацию посредством команды sendmail -bt, то покажется, будто локальная почта по-прежнему доставляется локально. Причина заключается в том, что директива nullclient обрабатывается позднее, в пятом наборе правил низкоуровневого файла конфигурации.

Клиент, на котором активизировано средство nullclient, должен иметь соответствующую запись MX, указывающую на сервер. Он должен быть также упомянут в файле /etc/mail/local-host-names сервера. Все эти установки позволят серверу принимать почту от данного клиента.

Сам клиент не должен принимать входящую почту. Если бы он это делал, ему все равно пришлось бы отсылать почту на сервер. Запустите программу sendmail без флага -bd, чтобы она не ожидала SMTP-запросов, поступающих через порт 25. Правда, это может нарушить работу некоторых пользовательских агентов, которые осуществляют начальную вставку сообщений в почтовую систему именно через порт 25. Более удачный способ отказаться от приема почты — запустить программу sendmail с флагом -bd, но с помощью параметра DAEMON_OPTIONS заставить ее прослушивать только интерфейс обратной связи. В любом случае оставьте флаг -q30m, чтобы при выходе почтового сервера из строя клиент мог помещать исходящие сообщения в очередь и отправлять их позднее.

Средство nullclient подходит для клиентских компьютеров небольшой организации, где есть центральный почтовый сервер. В крупных организациях на почтовый концентратор приходится слишком большая нагрузка. Лучше разделить серверы входящей и исходящей почты либо организовать иерархию почтовых серверов.

В SUSE есть образец mc-файла, в котором активизируется средство nullclient. Этот файл называется /etc/mail/linux.nullclient.mc. Укажите имя почтового сервера, запустите препроцессор m4 для создания файла sendmail.cf, и требуемая конфигурация готова.

Средства local_lmtp и smrsh

По умолчанию локальным агентом доставки почты пользователям и в файлы является утилита /bin/mail, а интерпретатор /bin/sh отвечает за доставку почты в программы. Но у программы sendmail теперь есть более удачные альтернативы, особенно что касается второго случая.

Если средство local_lmtp включено, то его аргументом является имя локального агента, который понимает протокол LMTP (Local Mail Transport Protocol — протокол доставки локальной почты), определенный в документе RFC2033. Стандартным средством доставки почты пользователям является программа mail.local, входящая в дистрибутив sendmail. Точно так же, средство smrsh задает путь к программе, которая используется для передачи почты на вход исполняемым файлам. Подробнее об утилитах mail.local и smrsh рассказывается в разделе 18.12.

Средство local_procmail

При включении средства local_procmail появляется возможность использовать в качестве локального агента доставки утилиту procmail. Средство принимает три аргумента: путь к исполняемому файлу procmail, массив аргументов для вызова утилиты и флаги агента доставки. Значения по умолчанию вполне приемлемы, лишь стандартный путь к файлу (/usr/local/bin/procmail) не соответствует установкам большинства дистрибутивов (/usr/bin).

Утилита procmail может делать гораздо больше, чем ее конкуренты /bin/mail и mail.local. Кроме доставки сообщений в почтовые ящики пользователей утилита спо-

собна сортировать сообщения по папкам, сохранять их в файлах, запускать программы и фильтровать спам. Конечно, она далеко не столь безопасна, как smrsh, но если вы не ограничиваете пользователей в выборе запускаемых программ (т.е. доверяете *всем* пользователям), то утилита procmail довольно удобна. Она не входит в дистрибутив sendmail, и если ее нет в системе, то ее можно загрузить с Web-сайта www.procmail.org.

С помощью средства local_procmail можно задать использование другой программы обработки почты вместо procmail. Для этого обманите программу sendmail, сделав вид, будто указываете путь к локальной копии утилиты procmail:

```
FEATURE(`local_procmail', `/usr/local/bin/mymailer')
```

Чтобы узнать статистику работы утилиты procmail, воспользуйтесь командой /usr/bin/mailstat (не путать с /usr/sbin/mailstats, которая выдает статистику работы программы sendmail). Она инсталлирована во всех наших дистрибутивах и обрабатывает журнальные файлы утилиты.

Макросы LOCAL_*

Если требуется написать несколько специальных правил для каких-то особых ситуаций, используйте набор макросов с префиксом LOCAL_. Примеры таких конструкций приведены далее в этой главе.

Конфигурационные опции

Конфигурационные опции и макроконстанты (команды O и D языка конфигурирования программы sendmail) могут быть заданы с помощью макроса define препроцессора m4. Полный список опций, доступных в виде переменных препроцессора, а также их значения по умолчанию приведены в файле cf/README. Для большинства случаев вполне подходят стандартные значения.

Вот несколько примеров:

```
define(`confTO_QUEUERETURN', `7d')
define(`confTO_QUEUEWARN', `4h')
define(`confPRIVACY_FLAGS', `noexpn')
```

Опция confTO_QUEUERETURN определяет, как долго может находиться в очереди сообщение, которое не удается доставить. Опция confTO_QUEUEWARN задает, сколько времени сообщение должно находиться в очереди, чтобы его отправитель получил уведомление о возникновении проблем с доставкой. В первых двух строках показанного выше примера устанавливаются значения 7 дней и 4 часа, соответственно.

В следующей строке запрещается SMTP-команда EXPN (раскрытие адреса). Опция confPRIVACY_FLAGS принимает разделенный запятыми список опций безопасности. При наличии запятых некоторые версии препроцессора m4 требуют ставить две пары кавычек (GNU-версия препроцессора, имеющаяся в Linux, от этого избавлена):

```
define(`confPRIVACY_FLAGS', ``noexpn, novrfy'')
```

📖 Более полная информация об опциях безопасности приведена в конце раздела 18.13.

Стандартные значения большинства опций приемлемы для организаций, в которых безопасность и производительность не являются критическими факторами. В частности, программа sendmail по умолчанию защищает от спама, отключая ретрансляцию,

а также требуя, чтобы адреса были полностью определенными и чтобы им соответствовали реальные IP-адреса. Если почтовый концентратор испытывает большую нагрузку и обрабатывает много списков рассылки, возможно, потребуется настроить ряд опций, влияющих на производительность.

Опции, которые приходится настраивать чаще всего, перечислены в табл. 18.11 (это лишь около 15% из почти 175 опций конфигурации). Там же указаны их значения по умолчанию. В целях экономии места префикс `conf` в названиях опций опущен. Например, опция `FALLBACK_MX` в действительности называется `confFALLBACK_MX`. Таблица разделена на части в соответствии с тем, для каких целей применяются те или иные опции: общих, связанных с ресурсами, производительностью, безопасностью/спамом и прочих. Некоторые опции попадают сразу в несколько категорий, однако мы указали их только в одном месте.

Таблица 18.11. Базовые опции конфигурации

	Опция	Описание и значение по умолчанию
Общие	`COPY_ERRORS_TO`	Адреса в заголовке "Cc" сообщений об ошибках (отсутствует)
	`DOUBLE_BOUNCE_ADDRESS`	Служит для перехвата спама; в некоторых системах спам направляется в файл `/dev/null`, но это чревато тем, что серьезные проблемы останутся незамеченными (`postmaster`)
Ресурсы	`MIN_FREE_BLOCKS`	Минимальное пространство в файловой системе для хранения поступающей почты (100)
	`MAX_MESSAGE_SIZE`	Максимальный размер в байтах одного сообщения (неограничен)
	`TO_`*имя*	Время тайм-аута для разных ситуаций (варьируется)
	`TO_IDENT`	Время ожидания запросов на аутентификацию отправителя; если задано значение 0, проверка не выполняется (`5s`)
	`MAX_DAEMON_CHILDREN`	Максимальное количество порожденных процессов[a] (не ограничено)
Производительность	`MCI_CACHE_SIZE`	Количество открытых кэшированных TCP-соединений (2)
	`MCI_CACHE_TIMEOUT`	Время, в течение которого кэшированные соединения остаются открытыми (`5m`)
	`HOST_STATUS_DIRECTORY`	См. описание в конце раздела 18.13 (значение по умолчанию отсутствует)
	`FALLBACK_MX`	См. описание в конце раздела 18.13 (значение по умолчанию отсутствует)
	`FAST_SPLIT`	Подавляет поиск записей MX при сортировке адресов получателей и распределении их по очередям; см. раздел 18.13 (1 — включено)
	`QUEUE_LA`	Показатель средней загруженности, при котором сообщения помещаются в очередь, а не отправляются немедленно (8*`число_процессоров`)
	`REFUSE_LA`	Показатель средней загруженности, при котором почта перестает приниматься (12*`число_процессоров`)
	`DELAY_LA`	Показатель средней загруженности, при котором доставка почты замедляется (0 — нет ограничений)
	`MIN_QUEUE_AGE`	Минимальный период времени, в течение которого сообщение должно находиться в очереди; позволяет улучшить обработку очереди на перегруженном компьютере (0)

Окончание табл. 18.11

	Опция	Описание и значение по умолчанию
Безопасность/спам	TRUSTED_USERS	Позволяет владельцам программ, управляющих списками рассылки, менять заголовок “From” и перестраивать базу псевдонимов (root, daemon, uucp)
	PRIVACY_FLAGS	Налагает ограничения на информацию, выдаваемую по протоколу SMTP (`authwarnings`)
	INPUT_MAIL_FILTERS	Список фильтров входящей почты (пуст)
	MAX_MIME_HEADER_LENGTH	Максимальный размер заголовков MIME (не ограничен)[б]
	CONNECTION_RATE_THROTTLE	Предотвращает атаки типа “отказ в обслуживании”, ограничивая скорость, при которой разрешается устанавливать почтовое соединение (не ограничена)
	MAX_RCPTS_PER_MESSAGE	Препятствует отправке спама; обработка избыточного числа получателей приостанавливается и генерируется временное сообщение об ошибке (число получателей не ограничено)
	DONT_BLAME_SENDMAIL	Отменяет ряд функций программы `sendmail`, связанных с безопасностью и проверкой файлов; не изменяйте значение этой опции без особой необходимости (`safe`)
	AUTH_MECHANISMS	Список механизмов аутентификации по протоколу SMTP для библиотеки Cyrus SASL[в]
Разное	LDAP_DEFAULT_SPEC	Спецификация базы данных LDAP, включая имя компьютера, на котором запущен сервер, и номер порта (не определена)

[а] Если быть более точным, то это максимальное количество дочерних процессов, выполняемых одновременно. При достижении предельного значения программа `sendmail` отказывается принимать запросы. Эта опция позволяет предотвратить атаки типа “отказ в обслуживании”.

[б] Эта опция защищает буфер пользовательского агента от переполнения. Рекомендуемое значение — “256/128”, что означает “256 байт на заголовок и 128 байт на каждый параметр заголовка”.

[в] Значение по умолчанию таково: “EXTERNAL GSSAPI KERBEROS_V4 DIGEST-MD5 CRAM-MD5”. Не добавляйте “PLAIN LOGIN”, т.к. это ведет к снижению безопасности.

18.10. Средства программы sendmail для борьбы со спамом

Спам — это жаргонное слово, обозначающее “мусорную”, навязчивую электронную почту коммерческого характера. Спам стал серьезной проблемой, в основном из-за того, что отправитель сообщения платит не за переданное количество байтов, а за факт подключения к сети Internet. Если же плата взимается за объем трафика, то создается одно сообщение, адресуемое тысяче получателей, но ретранслируемое через компьютер-посредник. При этом пользователь того компьютера платит за весь объем спама, а сам спамер оплачивает лишь одну копию. Естественно, пользователи во многих странах тоже весьма недовольны тем, что получают спам и вынуждены оплачивать его побайтово.

С точки зрения маркетологов, спам — очень полезная вещь. Эффективность его применения высока, накладные расходы, наоборот, низкие, а доставка осуществляется

мгновенно. Список, содержащий 30 миллионов электронных адресов, стоит всего около 40 долларов.

Многие спамеры пытаются изображать невинность, помещая в свое послание ссылку "remove" (удалить) и тем самым как бы предоставляя получателям возможность удалить себя из списка рассылки. На первый взгляд, все честно, но, отвечая на их письмо, получатель лишь подтверждает наличие действующего электронного адреса. Удалив получателя из одного списка, спамеры могут тут же внести его в другой список (они ведь не обещают сохранять конфиденциальность). Спамеры любят также менять заголовки своих сообщений, пытаясь замаскировать автора послания и адрес компьютера, с которого оно было отправлено.

Дельцы, продающие электронные адреса спамерам, недавно начали применять новый вид "словарной атаки" для поиска неизвестных ранее адресов. Берется список наиболее распространенных фамилий, и при помощи программы-сканера к ним добавляются различные инициалы в надежде случайно сформировать действительный адрес электронной почты. Для проверки адресов программа связывается с почтовыми серверами, скажем, пятидесяти крупных провайдеров и выполняет команду `VRFY` или `RCPT` для каждого из несметного числа образованных ею адресов.

Такое зондирование подвергает почтовый сервер огромной нагрузке и отрицательно сказывается на его способности обрабатывать почту "мирных" клиентов. Программа `sendmail` может бороться с этим, если для переменной `PrivacyOptions` задано значение `goaway` (см. табл. 18.15). Однако спамеры отличаются навязчивостью: если команда `VRFY` заблокирована, программа-сканер попытается выполнить команду `EXPN`, а если и это не удастся, то будет опробована команда `RCPT`. Таким способом опрашиваются миллионы адресов, и хотя не было послано ни одного сообщения, почтовый сервер гарантированно окажется загруженным. Для этих целей в арсенале программы `sendmail` предусмотрена специальная опция `BAD_RCPT_THROTTLE`. Когда число отвергнутых адресов в конверте сообщения превышает пороговое значение, программа начинает делать секундную паузу после каждого неудачного выполнения команды `RCPT`.

В программу `sendmail` включены достаточно эффективные средства борьбы со спамом и почтовыми вирусами. К сожалению, большинство провайдеров вынуждено сканировать всю почту, в результате чего опять-таки страдают ни в чем не повинные клиенты. И все же на компьютерах пользователей эти средства оказываются очень полезными.

Существуют четыре типа средств, предназначенных для борьбы со спамом.

- *Правила управления открытой ретрансляцией*, под которой понимается использование почтового сервера для отправки сообщения одним сторонним пользователем другому, тоже стороннему. Спамеры часто прибегают к ретрансляции, пытаясь таким способом замаскировать действительного адресата сообщения и тем самым избежать обнаружения со стороны своих собственных провайдеров. Это также позволяет им перекладывать финансовую и вычислительную нагрузку на чужие плечи.
- *База доступа*, позволяющая фильтровать почту согласно адресам, содержащимся в базе. Это напоминает применение брандмауэра для электронной почты.
- *Черные списки* открытых ретрансляторов и известных спамеров, проверяемые программой `sendmail`.
- *Проверка заголовков* и *фильтрация входящей почты* посредством универсальной библиотеки фильтров `libmilter`. Эта библиотека позволяет сканировать заголовки и содержимое сообщений и отбрасывать сообщения, удовлетворяющие определенным критериям.

Сначала мы опишем перечисленные средства, а затем познакомим с реальными примерами спама и покажем, как настроить почтовую систему, чтобы она распознавала и отвергала спам автоматически.

Ретрансляция

Программа `sendmail` и другие транспортные агенты получают входящую почту, просматривают адреса конверта, принимают решение о том, куда переслать сообщение, после чего отправляют его в соответствующий пункт назначения. Это может быть не только локальный получатель, но и другой транспортный агент, расположенный далее по цепи доставки. Когда входящее сообщение не имеет локального получателя, транспортный агент, обрабатывающий его, играет роль *ретранслятора* (relay).

До появления программы `sendmail` версии 8.9 по умолчанию включался режим "беспорядочной" рассылки (известный как *открытая ретрансляция*). Программа `sendmail` принимала сообщения через порт 25 и пыталась осуществить доставку оптимальным способом. В те времена считалось, что сеть Internet — дружественная среда.

К сожалению, ретрансляцией стали злоупотреблять спамеры. Они используют ее в целях маскировки и, что еще хуже, для перекладывания нагрузки (в том числе финансовой) на другие компьютеры. По этой причине конфигурирование почтового сервера в режиме открытой ретрансляции считается в настоящее время дурным тоном. Тем не менее, подобная конфигурация до сих пор встречается чрезвычайно широко.

Только хостам, помеченным ключевым словом `RELAY` в базе доступа, а также хостам, перечисленным в файле `/etc/mail/relay-domains`, разрешается предоставлять почту для ретрансляции. Вследствие изменений в стандартной конфигурации программы `sendmail` доля серверов открытой ретрансляции в течение ближайших лет должна снизиться. Начинает также сказываться осведомленность публики по этому вопросу и профилактическая деятельность организаций, публикующих черные списки.

Таким образом, ясно, что "беспорядочная" ретрансляция — это плохо. Но ясно и то, что некоторые типы ретрансляции необходимы. Как определить, какое сообщение следует ретранслировать, а какое — отклонить? Ретрансляция нужна лишь в двух случаях.

- *Когда транспортный агент выступает в качестве шлюза для хостов, недоступных никаким другим способом.* Это могут быть периодически выключаемые компьютеры (хосты с коммутируемым доступом, персональные компьютеры, работающие под управлением Windows) и виртуальные хосты. В этом случае все получатели, которым требуется переслать сообщение, должны находиться одном домене.
- *Когда транспортный агент является сервером исходящей почты для других хостов.* В этом случае имена и IP-адреса всех хостов-отправителей являются локальными (или, по крайней мере, заданы явно).

Любые другие случаи ретрансляции скорее всего свидетельствуют о плохой конфигурации сервера (исключение можно сделать в отношении мобильных пользователей). Чтобы избежать ретрансляции в первом из упомянутых вариантов, организуйте прием почты на центральном сервере (с использованием протокола POP или IMAP для клиентского доступа). Второй вариант должен быть всегда допустим, но только для локальных хостов. Лучше проверять IP-адреса, чем имена компьютеров, так как последние проще подделать; впрочем, программа `sendmail` следит за этим.

В программе `sendmail` ретрансляция запрещена по умолчанию, но с помощью специальных средств ее можно разрешить либо в полном объеме, либо в ограничен-

ном и контролируемом режиме. Эти средства перечислены ниже. Их можно использовать, но мы рекомендуем проявлять максимум осторожности и не злоупотреблять ими. Безопаснее всего организовать ограниченную ретрансляцию с помощью средства access_db, рассматриваемого в следующем подразделе.

- FEATURE(`relay_entire_domain') — разрешает ретрансляцию для хостов текущего домена;
- RELAY_DOMAIN(`*домен*,...') — дополнительно разрешает ретрансляцию для указанных доменов;
- RELAY_DOMAIN_FILE(`*имя_файла*') — то же самое, но список доменов берется из файла;
- FEATURE(`relay_hosts_only') — влияет на работу макроса RELAY_DOMAIN и средства access_db.

Если благодаря макроконстантам SMART_HOST и MAIL_HUB почта направляется на конкретный почтовый сервер, придется сделать исключение. Этот сервер должен быть настроен на ретрансляцию всей почты, поступающей от локальных компьютеров:

```
FEATURE(`relay_entire_domain')
```

Организациям, предоставляющим услуги виртуального хостинга, придется воспользоваться также макросом RELAY_DOMAIN, чтобы ретрансляция поддерживалась для виртуальных доменов, хотя инструкция

```
FEATURE(`use_cw_file')
```

выполняет, по сути, те же действия.

Ниже представлено еще несколько средств, использование которых связано с существенным риском.

- FEATURE(`promiscuous_relay') — разрешает "беспорядочную" ретрансляцию;
- FEATURE(`relay_based_on_MX') — разрешает ретрансляцию для всех хостов, записи MX которых ссылаются на текущий компьютер;
- FEATURE(`loose_relay_check') — разрешает "процентный" формат адресов;
- FEATURE(`relay_local_from') — разрешает ретрансляцию для локальных адресов, указанных в заголовке "From" сообщения.

Средство promiscuous_relay позволяет осуществлять пересылку почты из любого хоста на любой другой хост. Активизируя это средство, вы неминуемо попадете в черные списки. Не пользуйтесь этим средством на компьютере, который имеет выход в Internet.

Средство relay_based_on_MX опасно по той причине, что не позволяет контролировать, какие хосты ссылаются в своих записях MX на данный компьютер. Обычно соответствующие записи MX имеются только у локальных компьютеров, но ничто не мешает другим хостам поменять свои записи. Спамеры, как правило, лишены такой возможности, однако организации с сомнительной репутацией вольны в своем выборе.

Средство loose_relay_check разрешает применять "процентный" тип адресации (см. раздел 18.2), которым спамеры с удовольствием пользуются.

Средство relay_local_from заставляет программу sendmail доверять адресу отправителя, указанному в конверте сообщения, и осуществлять ретрансляцию тех сообщений, которые якобы поступают от локальных адресов. Естественно, и конверт, и заголовки почтовых сообщений подделать легко, а спамеры — большие специалисты по подделке.

Планируя разрешить ретрансляцию в той или иной форме, не поленитесь сначала обратиться к документации (файл cf/README), чтобы нечаянно не стать лучшим другом спамеров. После этого попросите специалистов проверить, не включен ли случайно режим открытой ретрансляции (попробуйте обратиться на хост spam.abuse.net).

При неправильной конфигурации компьютер может ретранслировать сообщения со странными адресами, основанными на искажении синтаксиса адресации UUCP. Чтобы предупредить появление такой "дыры", отключите протокол UUCP:

```
FEATURE(`nouucp', `reject')
```

В настоящее время программа sendmail по умолчанию не поддерживает древние сетевые технологии, такие как UUCP, BITNET и DECnet.

Еще один распространенный тип ретрансляции задается макроконстантой LUSER_RELAY. Это почта для локальных несуществующих пользователей:

```
define(`LUSER_RELAY', `error:No such user')
```

Хосты, на которых работает неправильно сконфигурированная программа sendmail, иногда допускают "утечку" сообщений с не полностью определенными именами локальных пользователей (обычно в поле "Cc"). Тот, кто попытается ответить на такое сообщение, на самом деле укажет в качестве адресата мнимого, несуществующего локального пользователя. Такая почта направляется агенту доставки error.

Менять эту конфигурацию имеет смысл лишь в том случае, если требуется модифицировать текст возвращаемого сообщения или реализовать особую обработку.

База доступа

Программа sendmail поддерживает базу доступа, которую можно использовать для создания почтового брандмауэра. База доступа будет проверять почту, приходящую извне, и отбраковывать сообщения, которые поступают от определенных пользователей или доменов. База доступа может также хранить имена доменов, для которых разрешается осуществлять ретрансляцию.

База доступа подключается следующим образом:

```
FEATURE(`access_db', `тип имя_файла')
```

Если аргументы *тип* и *имя_файла* не указаны, то по умолчанию принимаются тип hash (при условии, что используется СУБД Berkeley DB — это зависит от значения переменной DATABASE_MAP_TYPE) и файл /etc/mail/access. Базы данных формата DBM не используют аргумент *тип*. Как обычно, база доступа создается с помощью команды makemap:

```
# makemap hash /etc/mail/access < /etc/mail/access
```

Поле ключа в базе данных может содержать адреса электронной почты, имена пользователей, доменные имена и сетевые адреса. Например:

```
cyberspammer.com         550 Spam not accepted
okguy@cyberspammer.com   OK
badguy@aol.com           REJECT
sendmail.org             RELAY
128.32                   RELAY
170.201.180.16           REJECT
hotlivesex@              550 Spam not accepted
friend@                  550 You are not my friend!
```

Допустимые значения второго поля перечислены в табл. 18.12. Значение RELAY наиболее "либерально", если можно так сказать: в этом режиме сообщение принимается и направляется конечному получателю. В режиме OK сообщение принимается, но не ретранслируется. В режиме REJECT сообщение вообще не принимается. В режиме SKIP разрешается делать исключения. Например, если требуется ретранслировать почту всех хостов, кроме двух в центральном домене, укажите адреса этих хостов с атрибутом SKIP, а затем адрес домена — с атрибутом RELAY. Порядок записей в данном случае не важен.

Таблица 18.12. Допустимые значения второго поля базы доступа

Значение	Что означает
OK	Прием и доставка почты в обычном режиме
RELAY	Прием почты и пересылка ее получателю; разрешается ретрансляция на уровне отдельных хостов
SKIP	Позволяет делать исключения из более общих правил
REJECT	Отклонение почты с выдачей типового сообщения об ошибке
DISCARD	Удаление почты
FRIEND	Служит для борьбы со спамом, используется средством delay-checks: если адрес соответствует значению первого поля, другие правила проверки заголовков не применяются
HATER	Служит для борьбы со спамом, используется средством delay-checks: если адрес соответствует значению первого поля, применяются другие правила проверки заголовков
xxx сообщение	Выдача сообщения об ошибке; вместо *xxx* должен стоять числовой код, определенный в документе RFC821[a]
ERROR:*xxx сообщение*	То же, что и предыдущее, но сообщение об ошибке помечается более явно
ERROR:*x.x.x сообщение*	То же, что и предыдущее; вместо *x.x.x* должен стоять один из кодов состояния доставки, определенных в документе RFC1893

[a] Например, 550 — почтовый ящик недоступен.

Приведенная выше база доступа пропускает сообщения от пользователя okguy@cyberspammer.com, но отклоняет все остальные сообщения от этого хоста, выдавая указанное сообщение об ошибке. Почта, приходящая из домена sendmail.org или сети 128.32.0.0/16 (сеть Калифорнийского университета в Беркли), будет ретранслироваться. Почта от пользователя badguy@aol.com будет отклоняться, как и почта, приходящая от пользователей hotlivesex и friend любых доменов.

В поле ключа могут стоять адреса IPv6 с разделителями в виде двоеточий, но эти адреса должны помечаться префиксом IPv6:. Символ @ после имен hotlivesex и friend необходим для того, чтобы отличать имена пользователей от имен доменов.

Код ошибки 550 описан в документе RFC821. Кодов состояния доставки, определенных в документе RFC1893, гораздо больше. Первая цифра 4 означает временную ошибку, цифра 5 — постоянную ошибку. Несколько кодов приведено в табл. 18.13.

Таблица 18.13. Коды состояния доставки (документ RFC1893)

Временная ошибка	Постоянная ошибка	Значение
4.2.1	5.2.1	Почтовый ящик отключен
4.2.2	5.2.2	Почтовый ящик переполнен
4.2.3	5.2.3	Сообщение слишком велико
4.2.4	5.2.4	Проблема с раскрытием списка рассылки
4.3.1	5.3.1	Почтовая система переполнена
4.4.4	5.4.4	Маршрутизация невозможна
4.4.5	5.4.5	Перегрузка почтовой системы
4.7.*	5.7.*	Нарушение правил безопасности

Поле ключа может содержать дескрипторы Connect, To, From и (в версии 8.12 и выше) Spam, позволяющие задавать еще более точные критерии фильтрации. Дескриптор Connect содержит информацию о соединении, например IP-адрес или имя клиента. Дескрипторы To и From ссылаются на адреса конверта, а не заголовков. Дескриптор Spam позволяет делать исключения из более общих правил благодаря специальным проверкам. Эти проверки включаются с помощью средства `delay_checks`:

```
FEATURE('delay_checks', 'friend')
FEATURE('delay_checks', 'hater')
```

В первом случае, если адрес содержится в базе доступа с пометкой `FRIEND`, то к сообщению не применяются другие правила, которые могут привести к тому, что сообщение не будет принято. Во втором случае, если адрес содержится в базе доступа с пометкой `HATER`, то дополнительные правила применяются. Совместно перечисленные четыре дескриптора обеспечивают улучшенный контроль над ретрансляцией и фильтрацией почты и позволяют переопределять другие установки. Если отдельные пользователи жалуются на то, что правила фильтрации спама мешают им работать, подкорректируйте правила с помощью ключевых слов `FRIEND` и `HATER`.

При наличии любого из этих дескрипторов поиск сначала выполняется по условию, заданному дескриптором, а затем — обычным способом, что необходимо для поддержания обратной совместимости с более старыми базами доступа.

Вот несколько примеров:

```
From:spammer@some.domain      REJECT
To:friend.domain              RELAY
Connect:friend.domain         OK
Spam:abuse@                   FRIEND
```

Почта, приходящая от имени `spammer@some.domain`, будет заблокирована, но по этому адресу все равно можно будет посылать почту, даже если он помещен в черный список. Разрешается ретранслировать почту, посылаемую по адресу `friend.domain`, но это не относится к сообщениям, поступающим от данного адреса (предполагается, что ретрансляция запрещена где-то в другом месте). Соединение с хостом `friend.domain` допускается, даже если этот адрес занесен в один из черных списков DNS. Почта, адресуемая пользователю `abuse@`*`локальный_домен`*, будет проходить, даже если она поступает из адреса `spammer@some.domain`, сообщения которого отвергаются согласно первому правилу.

Базы доступа используются многими организациями с целью контроля спама. Наш главный сервер входящей почты на факультете вычислительной техники университета штата Колорадо использует средство `access_db` для отклонения почты от более чем 500 известных спамеров, идентифицируемых по адресам, доменам или IP-сетям.

Черные списки

Если необходимо блокировать почту каких-либо локальных пользователей или хостов, воспользуйтесь средством

```
FEATURE(`blacklist_recipients')
```

которому в базе доступа соответствуют записи следующих типов:

```
To:nobody@                  550 Mailbox disabled for this user
To:printer.mydomain.edu     550 This host does not accept mail
To:user@host.mydomain.edu   550 Mailbox disabled for this user
```

Эти записи блокируют почту, приходящую на адрес пользователя nobody (на любом хосте), а также почту, адресованную сетевому принтеру и одному из пользователей конкретного компьютера. Дескриптор `To:` разрешает этим пользователям посылать сообщения (подобной возможностью обладают некоторые принтеры), но не принимать их.

К сожалению, сегодня практически невозможно поддерживать подобные черные списки вручную. К счастью, через DNS можно получить доступ к черным спискам, поддерживаемым сообществом пользователей, причем некоторыми из них можно пользоваться совершенно бесплатно.

С помощью средства `dnsbl` можно подключить DNS-списки блокируемых адресов:

```
FEATURE(`dnsbl', 'sbl-xbl.spamhaus.org')
```

Это средство заставляет программу `sendmail` отклонять почту, приходящую от хоста, IP-адрес которого имеется в списке известных спамеров Spamhaus Block List, который поддерживает группа `sbl-xbl.spamhaus.org`. В других списках указаны сайты, поддерживающие открытую ретрансляцию, и известные блоки адресов коммутируемого доступа, которые могут быть "прибежищем" для спамеров.

Черные списки распространяются через службу DNS с помощью особого приема. Например, специальная DNS-запись

```
IP-адрес.blackholes.mail-abuse.org  IN  A  127.0.0.2
```

введенная в базу данных DNS домена `blackholes.mail-abuse.org`, блокирует почту от указанного хоста, если средство `dnsbl` включено (программа `sendmail` явно проверяет, существует ли такая запись). В данном примере IP-адрес — это сетевой адрес в обратной точечной нотации.

Средство `dnsbl` можно активизировать несколько раз, чтобы проверить различные списки злоумышленников. Достаточно добавить второй аргумент, задающий сервер имен для черного списка, и третий аргумент, содержащий требуемое сообщение об ошибке. Если третий аргумент опущен, будет выдано фиксированное сообщение (из базы данных DNS, содержащей проверяемые записи).

Проверка заголовков

Спамеры часто пытаются скрыть свою личность. С момента появления программы `sendmail` версии 8.9 введено ограничение: если адрес "From" конверта не представлен в

виде *пользователь@существующий_домен*, почта не принимается. От этого можно отказаться с помощью следующих средств:

```
FEATURE(`accept_unresolvable_domains')
FEATURE(`accept_unqualified_senders')
```

Первое из них разрешает программе sendmail принимать почту от доменов, которые не существуют или не распознаются в DNS. Второе разрешает, чтобы адреса в заголовке "From" содержали только имя пользователя без указания доменной части. Использовать эти средства рекомендуется лишь в том случае, когда сеть организации защищена брандмауэром, а в базе данных DNS присутствуют сугубо локальные данные. В противном случае подумайте лучше о том, чтобы изменить структуру почтовой системы и службы DNS. Требование, чтобы адрес отправителя, указанный в конверте, был реальным, способствует существенному сокращению спама.

Проверка заголовков — это мощный механизм борьбы со спамом, в основе которого лежит использование синтаксиса низкоуровневого файла конфигурации программы sendmail. Проверяя заголовки, программа сверяет их с имеющимися образцами (например, "To: friend@public.com") и отклоняет подозрительные сообщения до того, как они будут доставлены в почтовые ящики пользователей.

Проверка заголовков может также применяться для распознавания почтовых вирусов при условии, что они имеют характерную строку заголовка. Например, вирус Melissa (1999 г.) содержал заголовок "Subject" вида "Important Message From...". Через несколько часов после обнаружения вируса Melissa на хосте www.sendmail.com был опубликован набор локальных правил по идентификации вируса и фильтрации содержащих его сообщений.

То же самое происходит и в отношении других почтовых вирусов: как только выясняются отличительные особенности вируса и появляется возможность выразить их в виде правил программы sendmail, быстро публикуются исправления к почтовой системе (как на Web-сайте www.sendmail.com, так и на хосте www.sendmail.org).

Характерный образец правил фильтрации, предназначенных для борьбы со спамом и вирусами, можно найти в файле конфигурации программы sendmail для домашнего компьютера Эрика Оллмана (knecht). Этот файл включен в дистрибутив sendmail (cf/cf/knecht.mc). Скопируйте из него правила фильтрации спама и добавьте их в конец своего mc-файла.

Анализируя различные примеры, мы встречали следующие правила проверки заголовков. Проверяется:

- почта, адресованная любому пользователю в домене public.com;
- почта, адресованная пользователям "friend" и "you";
- почта, имеющая заголовок "X-Spanska", который указывает на наличие вируса-червя Happy99;
- почта с заголовком "Subject" вида "Important Message From..." (вирус Melissa);
- почта с заголовком "Subject" вида "all.net and Fred Cohen..." (вирус Papa);
- почта с заголовком "Subject" вида "ILOVEYOU" (вирус "iloveyou" и его варианты);
- почта с искаженным заголовком Outlook Express (вирус-червь SirCam);
- сотни видов рекламного мусора.

Все правила проверки заголовков задаются инструкциями LOCAL_CONFIG и LOCAL_RULESETS, расположенными в конце mc-файла. С помощью команды divert препро-

цессора m4 программа sendmail размещает эти правила в нужном месте низкоуровневого файла конфигурации.

В ответ на получение спама вызывайте агент доставки error (с сообщением об ошибке вида "user unknown"), а не discard. Многие спамеры "подчищают" свои списки рассылки, оставляя только коммерчески ценные адреса, поэтому иногда можно удалить себя из списка, записав в него сообщение об ошибке.

Ограничение скорости и количества соединений

Версия 8.13 программы sendmail содержит средство ratecontrol, которое показывает, сколько входящих соединений может быть произведено с данного хоста или сети. Это ограничение может быть полезным в частности для того, чтобы ослабить поток спама, поступающий из источников, которые невозможно полностью заблокировать без негативных последствий (это может быть полезно, например, для крупного поставщика услуг Internet, у которого зарегистрировано большое количество пользователей, применяющих коммутируемый доступ). Чтобы управлять скоростью, нужно написать в файле .mc следующую строку:

```
FEATURE(`ratecontrol', `nodelay', `terminate')
```

Кроме того, вы должны перечислить хосты или сети, за которыми нужно вести наблюдение, а в файле /etc/mail/access указать пределы ограничений. Например, строки

```
ClientRate:192.168.6.17 2
ClientRate:170.65.3.4 10
```

разрешают компьютерам 192.168.6.17 и 170.65.3.4 два и десять новых соединений в течение одной минуты, соответственно.

Средство conncontrol ограничивает количество одновременных соединений. Его можно активировать в файле .mc следующим образом:

```
FEATURE(`conncontrol', `no delay',`terminate')
```

Как и для средства ratecontrol, хосты и сети определяются в файле /etc/mail/access:

```
ClientConn:192.168.2.8 2
ClientConn:175.14.4.1 7
ClientConn: 10
```

Эта конфигурация расшифровывается так: хост 192.168.2.8 может иметь два одновременных соединения, хост 175.14.4.1 — семь одновременных соединений, все остальные хосты — по десять одновременных соединений.

"Хлопанье дверью"

Еще одним искусным средством в версии 8.13 является greet_pause. Когда дистанционный почтовый агент соединяется с вашим сервером sendmail, протокол SMTP поручает, чтобы агент приостановил свои действия до тех пор, пока ваш сервер не отреагирует приветствием. Однако спам-почтальоны (а также черви/вирусы) могут сразу же выдавать команду EHLO/HELO. Это поведение может быть вызвано или плохой реализацией протокола SMTP в средствах рассылки спама, или действием программы, целью которой является задержка времени по поручению спамера. В любом случае, это поведение является опасным и называется "хлопаньем дверью" (slamming).

Средство greet_pause заставляет программу sendmail ждать в течение определенного периода времени в начале соединения, прежде чем приветствовать нового "приятеля". Если дистанционный почтовый агент не бездействует в течение этого времени, а выдает команду EHLO или HELO, sendmail регистрирует ошибку и отвергает команды, поступающие от дистанционного почтового агента.

Паузу на время приветствия можно активировать с помощью следующей записи в файле .mc:

```
FEATURE(`greet_pause', `700')
```

Эта строка устанавливает задержку, равную 700 миллисекундам, в начале каждого нового соединения. Можно установить задержки для хоста или сети, подобно средствам conncontrol и ratecontrol, однако в большинстве организаций для этого средства используется указанное выше общее значение.

Фильтрация почты

В программу sendmail версии 8.12 включен обобщенный механизм фильтрации заголовков, который со временем может превратиться в очень эффективный инструмент борьбы со спамом. Это библиотека функций фильтрации почты, позволяющая пользователям создавать собственные фильтры. С помощью фильтров можно выявлять во входящей почте признаки вирусов или спама и удалять либо регистрировать такие сообщения. Возможна проверка как заголовков, так и содержимого сообщений.

Фильтрация сообщений — мощное средство не только борьбы со спамом, но и нарушения конфиденциальности. В последнем качестве оно может представлять интерес для менеджеров, которые хотят знать, какая информация передается по электронной почте пользователями компании. Фильтрация исходящей почты не поддерживается в версии 8.12, но анонсирована как одна из первоочередных задач.

Библиотека функций фильтрации называется libmilter. Она активизируется с помощью конфигурационных директив INPUT_MAIL_FILTER и MAIL_FILTER и контролируется опциями MILTER_MACROS_*, которые позволяют указывать, какие фильтры должны применяться на той или иной стадии SMTP-диалога.

Например, инструкция

```
INPUT_MAIL_FILTER(`filtername',
                  `S=mailer:/var/run/filtername.socket')
```

перенаправляет все входящие сообщения программе /etc/mail/filtername через сокет, указанный во втором аргументе.

Дополнительную информацию по данной теме можно найти в файле libmilter/README и в каталоге libmilter/docs дистрибутива sendmail. В файле README показан пример простейшего фильтра, регистрирующего сообщения в файле. В HTML-файлах каталога docs описан интерфейс библиотеки и объясняется применение различных функций в пользовательских программах фильтрации.

Обработка спама

Борьба со спамом может оказаться трудным и часто безрезультатным занятием. Не стоит тратить усилия на поимку отдельных спамеров, даже если многие из них сумеют прорваться через заградительные фильтры. Время, потраченное на анализ заголовков спама и прочие переживания, — потерянное время. Да, это благородная борьба, но объем поступающего спама она вряд ли уменьшит.

Постоянных спамеров можно нейтрализовать довольно быстро, вычислив их провайдера и послав ему жалобу. Однако спамеры, работающие по принципу "напакостил и удрал", используют учетную запись на хосте провайдера только один раз, после чего избавляются от нее. Их поймать трудно. Если они рекламируют свой Web-сайт, то ответственность можно возложить на него. Если же они объявляют номер телефона или почтовый адрес, то идентифицировать отправителя спама тяжело, хотя, в принципе, возможно. "Мобильные" спамеры могут оказаться вообще недосягаемыми для наказания.

Всевозможные черные списки в некоторой степени эффективны для блокировки спама и приводят к уменьшению числа серверов, поддерживающих открытую ретрансляцию. Оказаться в таком списке — значит подорвать свою деловую репутацию, поэтому многие провайдеры и организации внимательно следят за действиями своих пользователей.

Наш главный совет, касающийся спама, таков: регулярно проводите профилактические мероприятия и пользуйтесь имеющимися черными списками. Другой вариант: перенаправляйте вашу входящую почту в стороннюю компанию, занимающуюся борьбой со спамом (например, Postini, `www.postini.com`). С другой стороны, в последнем случае может пострадать производительность, секретность или надежность.

Посоветуйте своим пользователям просто удалять полученный ими спам. Многие сообщения содержат инструкции о том, как получатель может удалить себя из списка рассылки. Не попадайтесь на эту "грязную" уловку. Если последовать таким инструкциям, то спамеры действительно могут удалить получателя из текущего списка, но при этом они немедленно внесут его в несколько других списков, причем сопроводят адрес аннотацией вида "Вот реальный клиент, который читает наши сообщения". После этого, как вы понимаете, ценность электронного адреса возрастет, а вместе с ним — и объем получаемого спама.

Если вам все же нравится вести борьбу со спамом, воспользуйтесь помощью, предлагаемой соответствующими Web-сайтами. В первую очередь стоит упомянуть хосты `www.abuse.net`, `www.spamcop.net` и `www.cauce.org`. SpamCop предлагает средства анализа заголовков почтовых сообщений и определения действительного отправителя. На сайте cauce.org можно найти полезную информацию о законах в отношении рассылки спама. В США можно обратиться в Федеральную торговую комиссию (Federal Trade Commission). Их сайт находится по адресу `www.ftc.gov/spam`.

SpamAssassin

SpamAssassin — это фильтр (его можно вызывать посредством программы `sendmail`), который очень эффективно обнаруживает спам. Он использует систему баллов для определения степени фальшивости сообщения. Этот фильтр редко допускает ошибки и перехватывает все сообщения, действительно являющиеся спамом. Если сообщение набирает слишком много баллов (конкретное значение конфигурируется на основе данных о сайте и пользователе), программа отмечает его с помощью метки. После этого вы сможете переместить подозрительные сообщения в папку со спамом, или запустив фильтр на стороне сервера (например, Cyrus `sieve`), или сконфигурировав свой пользовательский агент. Вы можете даже научить SpamAssassin тому, какие письма являются "здоровыми", а какие — нет. Прежде чем настраивать параметры SpamAssassin, внимательно изучите характеристики всех нежелательных писем. Эта программа доступна на Web-сайте `spamassassin.apache.org`.

SPF и Sender ID

Лучший способ борьбы со спамом — остановить его распространение на самом источнике. На первый взгляд, сделать это можно легко и просто, однако на самом деле это практически неразрешимая задача. Дело в том, что сама структура сети Internet сильно усложняет отслеживание настоящего источника сообщения и проверку его подлинности. Обществу нужен верный способ проверки того, что письмо было действительно отправлено тем пользователем, который указан в заголовке From.

Эту задачу пытались решить многими способами, однако наибольший эффект был достигнут благодаря использованию SPF (Sender Policy Framework — система политики отправителя) и Sender ID (Идентификатор отправителя). SPF сейчас существует в качестве стандарта (RFC4408). Она определяет набор записей DNS TXT (см. раздел 15.7), посредством которых организация может идентифицировать свои "официальные" внешние ретрансляторы почты. На основании этого почтовые агенты могут отклонять любое электронное письмо, поступившее из домена данной организации, если письмо не было отправлено из данных официальных источников. Естественно, эта схему будет работать только в том случае, если множество организаций опубликуют записи SPF. В программе `sendmail` эти функции могут быть реализованы некоторыми фильтрами, доступными в сети Internet.

Sender ID и SPF практически идентичны по форме и выполняемым функциям. Однако ключевые части Sender ID запатентованы компанией Microsoft и поэтому являются предметом горячих дискуссий. На момент написания этой книги Microsoft попрежнему пыталась заставить всех и вся принять принадлежащие ей стандарты.

18.11. Примеры файлов конфигурации

В этом разделе мы проанализируем конфигурационные файлы небольшой компании Sendmail, Inc., администраторы которой знают, как правильно настраивать программу `sendmail`. Их почтовая система включает в себя главный почтовый концентратор для входящих и исходящих сообщений. Вся входящая почта принимается и немедленно пересылается группе внутренних серверов IMAP, которые проверяют каждое сообщение на предмет наличия вирусов, прежде чем доставлять его в почтовый ящик пользователя. Аналогичная проверка исходящей почты осуществляется на хосте-концентраторе, так что никто не сможет обвинить компанию в распространении вирусов по электронной почте. Сначала мы познакомимся с конфигурацией клиентов, а затем изучим более сложную конфигурацию главного сервера. В примерах мы слегка изменили оригинальные файлы: исключили примечания о защите авторских прав, добавили ряд комментариев и удалили макросы `dnl` в конце строк.

Клиентские компьютеры на хосте sendmail.com

Файл `smi-client.mc` для клиентских компьютеров довольно прост. В нем содержится ссылка на сервер smtp.sendmail.com, хотя в действительности это псевдоним хоста `foon.sendmail.com`. Применение записи MX (или CNAME)[31] в качестве ссылки на почтовый сервер удобно тем, что ее легко изменить при смене сервера.

[31] Запись MX эффективнее, чем CNAME, так как последняя требует дополнительной операции поиска имени для определения IP-адреса.

Обратите внимание на то, что файл датирован октябрем 1998 года. С тех пор программа sendmail неоднократно обновлялась, однако конфигурационный файл не пришлось менять.

```
divert(-1)
##### Этот файл содержит настройки для клиентских компьютеров
##### компании Sendmail, Inc.
divert(0)
VERSIONID(`@(#)smi-client.mc 1.0 (Sendmail) 10/14/98')
OSTYPE(`bsd4.4')
FEATURE(`nocanonify')
undefine(`ALIAS_FILE')
define(`MAIL_HUB', `smtp.sendmail.com')
define(`SMART_HOST', `smtp.sendmail.com')
define(`confFORWARD_PATH', `')
MAILER(`local')
MAILER(`smtp')
```

Макроконстанты MAIL_HUB и SMART_HOST задают перенаправление входящей и исходящей почты на хост smtp.sendmail.com. Записи MX базы данных DNS должны указывать на то, что этот хост имеет более высокий приоритет (меньший номер записи), чем отдельные клиентские компьютеры. Путь к файлам .forward устанавливается пустым, файл псевдонимов также пуст. Раскрытие всех псевдонимов осуществляется на главном сервере. Средство nocanonify включено здесь для повышения производительности, поскольку поиск в DNS тоже выполняется на главном сервере.

Главный сервер хоста sendmail.com

Главный почтовый сервер на хосте sendmail.com подвергается, наверное, наиболее массированным атакам извне среди всех известных нам инсталляций программы sendmail. Он должен быть устойчивым ко всякого рода почтовым атакам и защищать клиентские компьютеры. Вот файл конфигурации сервера:

```
divert(-1)
# Создан программой Sendmail Switch, коммерческим продуктом
# компании sendmail.com.
divert(0)
ifdef(`COMMERCIAL_CONFIG', `INPUT_MAIL_FILTER(`mime-filter',
     `S=local: /var/run/mime-filter/mime-filter.sock')')
LDAPROUTE_DOMAIN(`sendmail.com sendmail.net sendmail.org')
MASQUERADE_AS(`sendmail.com')
MASQUERADE_DOMAIN(`sendmail.com')
RELAY_DOMAIN(`sendmail.com sendmail.net sendmail.org')
define(`MAIL_HUB', `internal-hub.sendmail.com')
define(`QUEUE_DIR', `/var/spool/mqueue/q*')
define(`SMART_HOST', `virus-scan.sendmail.com')
ifdef(`COMMERCIAL_CONFIG', `define(`confCACERT',
     `/local/certs/cacert.pem')')
ifdef(`COMMERCIAL_CONFIG', `define(`confCACERT_PATH',
     `/local/certs/trustedcerts')')
define(`confCHECK_ALIASES', `True')
ifdef(`COMMERCIAL_CONFIG', `define(`confCLIENT_CERT',
     `/local/certs/cert.pem')')
```

```
ifdef(`COMMERCIAL_CONFIG', `define(`confCLIENT_KEY',
     `/local/certs/key.pem')')
define(`confEIGHT_BIT_HANDLING', `mimify')
define(`confLDAP_DEFAULT_SPEC', ` -h "ldap.sendmail.com
      ldap2.sendmail.com" -b "dc=sendmail,dc=com" -p 1389')
define(`confREFUSE_LA', `99')
define(`confRUN_AS_USER', `mailnull')
ifdef(`COMMERCIAL_CONFIG', `define(`confSERVER_CERT',
     `/local/certs/cert.pem')')
ifdef(`COMMERCIAL_CONFIG', `define(`confSERVER_KEY',
     `/local/certs/key.pem')')
define(`confTO_IDENT', `0s')
define(`confTO_QUEUEWARN', `2d')
ifdef(`confPOP_TO', `', `define(`confPOP_TO', `900')')
FEATURE(`accept_unqualified_senders')
FEATURE(`accept_unresolvable_domains')
FEATURE(`allmasquerade')
FEATURE(`always_add_domain')
FEATURE(`domaintable')
FEATURE(`ldap_routing', `ldap -1 -v mailHost -k ldap -1 -v mailhost
     -k (&(objectclass=mailRecipient)(|(mail=%0)
         (|(mailAlternateAddress=%0))))',
     `ldap -1 -v mail -k
     (&(objectclass=mailRecipient)(|(mailAlternateAddress=%0)))',
     `passthru')
FEATURE(`mailertable')
FEATURE(`masquerade_entire_domain')
FEATURE(`masquerade_envelope')
FEATURE(`relay_entire_domain')
FEATURE(`use_cw_file')
MAILER(`local')
MAILER(`smtp')

LOCAL_RULESETS
SLocal_check_rcpt
R$*    $: $&{verify}
ROK    $# OK
```

Задача главного сервера заключается в распределении входящей почты между внутренними серверами и ретрансляции исходящей почты в режиме `SMART_HOST`. Следующие две строки

```
FEATURE(`accept_unqualified_senders')
FEATURE(`accept_unresolvable_domains')
```

обеспечивают прием поступающей почты, даже если имя отправителя неполное или указанный в адресе домен не распознается. Это позволяет обслуживать почту потенциальных клиентов, у которых неправильно сконфигурирована программа `sendmail` либо служба DNS. Тем самым, однако, отменяются стандартные правила фильтрации спама и прочих сообщений с искаженными заголовками. Отключена также идентификация отправителя (тайм-аут равен нулю), что ускоряет доставку входящей почты.

В первую очередь сервер проверяет сообщения с определенными типами MIME-вложений, в которых часто прячутся вирусы (инструкция `INPUT_MAIL_FILTER`). Вызываемый при этом сценарий `mime-filter` содержит строки следующего вида:

```
*:anniv.doc:*  error:Your email was not accepted by Sendmail,
    it appears to be infected with the Melissa-X virus.
*:.vbs:*      error:For security and virus protection reasons,
    Sendmail does not accept messages with VBS files attached.
    Please retransmit your message without the VBS file.
```

MIME-вложения с расширениями `.vba`, `.dot`, `.exe`, `.com`, `.reg` и т.д. не принимаются, хотя полное сканирование на предмет наличия вирусов здесь не проводится, так как это замедлило бы обработку входящей почты. Вместо этого в базе данных LDAP выполняется поиск получателя каждого такого сообщения, которое затем пересылается соответствующему внутреннему серверу IMAP/POP. Если получатель не найден, почта направляется главному внутреннему серверу (определяется значением переменной `MAIL_HUB`) для дальнейшей обработки. Внутренние серверы IMAP/POP и главный внутренний сервер осуществляют полное сканирование сообщений, прежде чем доставлять их в почтовые ящики пользователей.

Исходящая почта тоже проходит через главный почтовый сервер благодаря наличию установки `SMART_HOST` в файлах конфигурации клиентов. Если хосты, находящиеся за пределами домена sendmail.com, хотят переслать почту через его почтовые серверы, они должны предоставить сертификат, подписанный центром сертификации хоста sendmail.com. Такая возможность предусмотрена только для служащих компании, которые выезжают к заказчикам. Аутентификация каждого пользователя не позволяет поддельной почте проходить через хост sendmail.com.

Приняв почту, отправляемую в Internet, главный сервер передает ее сканирующему серверу, адрес которого указан в переменной `SMART_HOST`. В принципе, загруженность главного сервера позволяет ему самостоятельно осуществлять подобное сканирование, но тогда при отправке почты пользователям приходилось бы дожидаться окончания сканирования. Вместо этого почта помещается в очередь на другом компьютере, и пользователи счастливы: им кажется, будто почта отправляется мгновенно.

В конце файла задаются правила проверки заголовков на предмет наличия в них признаков различных вирусов и адресов известных спамеров. Хорошие образцы правил можно найти в файле `knecht.mc` из дистрибутива `sendmail`. Ниже мы приведем фрагмент этого файла.

Летом 2001 г. в сети Internet свирепствовал разрушительный вирус-червь SirCam. Показанный ниже фрагмент файла `knecht.mc` позволяет выловить этот вирус. SirCam — один из первых почтовых вирусов с произвольными заголовками. Обычными средствами его нельзя было поймать, но "помогли" сами авторы вируса, допустившие ошибку, вследствие которой сообщение вируса отличается от настоящего сообщения программы Outlook Express. Текст сообщения неизменен (в нем содержится просьба взглянуть на присоединенный документ), поэтому его вполне можно выявлять с помощью функций новой библиотеки `libmilter`, появившейся в версии 8.12. Сражаясь со всеми этими вирусами и червями, нацеленными на продукты Microsoft, невольно задумываешься над тем, чтобы "снести" все Windows-системы и поставить вместо них Linux!

📖 О библиотеке `libmilter` рассказывается в разделе 18.12.

```
LOCAL_RULESETS

KSirCamWormMarker regex -f -aSUSPECT multipart/mixed;boundary=----
    .+_Outlook_Express_message_boundary
HContent-Type: $>CheckContentType
```

```
SCheckContentType
R$+ $:          $(SirCamWormMarker $1 $)
RSUSPECT        $#error $: "553 Possible virus, see http:
  //www.symantec.com/avcenter/venc/data/w32.sircam.worm@mm.html"

HContent-Disposition:$>CheckContentDisposition

SCheckContentDisposition
R$-             $@ OK
R$- ;           $+ $@ OK
R$*             $#error $: "553 Illegal Content-Disposition"
```

В конфигурационных файлах клиентских компьютеров не контролируется спам, так как вся поступающая на хост sendmail.com почта проходит через внешний и внутренний концентраторы, где и происходит фильтрация. Некоторые средства и конструкции, использованные в этом примере, не описаны в книге, но необходимую информацию можно найти в файле `cf/README`.

18.12. Безопасность и программа sendmail

С началом лавинообразного роста сети Internet программы наподобие `sendmail`, принимающие произвольные входные данные и направляющие их локальным пользователям, в файлы или в интерпретаторы команд, превратились в излюбленные мишени для хакеров. В настоящее время программа `sendmail` наряду со службой DNS и даже протоколом IP начала обзаводиться встроенными средствами аутентификации и шифрования, позволяющими решать основные проблемы безопасности.

Недавнее послабление федеральных законов США, касающихся экспорта систем шифрования, позволило оснастить программу `sendmail` собственными механизмами шифрования. Версии 8.11 и выше поддерживают систему SMTP-аутентификации и SSL (Secure Sockets Layer — протокол защищенных сокетов). Протокол SSL в `sendmail` называется TLS (Transport Layer Security — безопасность транспортного уровня) и реализован в виде дополнения STARTTLS к протоколу SMTP. Ему соответствуют шесть новых конфигурационных параметров, связанных с файлами сертификатов и ключей. В базе доступа появились новые дескрипторы, указывающие на необходимость успешной аутентификации.

В этом разделе мы опишем эволюцию модели безопасности программы `sendmail`. В конце будут кратко рассмотрены протоколы TLS и SASL (Simple Authentication and Security Layer — простой протокол аутентификации и защиты) и их применение.

Безопасность программы `sendmail` постоянно повышалась, и теперь программа, прежде чем довериться содержимому, скажем, файлов `.forward` или `aliases`, тщательно проверяет права доступа к ним. Как правило, подобные меры предосторожности действительно необходимы, но иногда все же требуется их ослабить. С этой целью в программе появилась опция `DontBlameSendmail`, название которой (“не обвиняй `sendmail`”) подсказывает системным администраторам, что после изменения опции действия программы станут небезопасными.

У опции `DontBlameSendmail` очень много возможных значений (примерно 55).

По умолчанию она равна `safe`. Полный список значений можно найти в файле `doc/op/op.ps` дистрибутива `sendmail`.

Владельцы файлов

Программа sendmail различает трех специальных пользователей: DefaultUser, TrustedUser и RunAsUser.

По умолчанию все агенты доставки работают от имени пользователя DefaultUser, если не установлены специальные флаги. При наличии в файле /etc/passwd учетной записи mailnull, sendmail или daemon именно эта запись будет соответствовать пользователю DefaultUser. В противном случае пользователю будут сопоставлены значения UID и GID, равные 1. Мы рекомендуем применять учетную запись mailnull. Добавьте ее в файл /etc/passwd со звездочкой в поле пароля, с пустым полем идентификатора команд, с пустым полем начального каталога и группой mailnull. Понадобится также добавить запись mailnull в файл /etc/group. Этой учетной записи не должны принадлежать никакие файлы. Если программа sendmail не выполняется с правами суперпользователя, для агентов доставки должен быть установлен бит setuid.

Когда задан пользователь RunAsUser, программа sendmail работает от его имени, игнорируя пользователя DefaultUser. Если программа запускается с установленным битом setgid (идентификатор группы меняется на smmsp), то агент подачи почты (программа sendmail) передает транспортному агенту (другой экземпляр sendmail) сообщения по протоколу SMTP. Транспортный агент sendmail не имеет установленного бита setuid, но вызывается с правами суперпользователя из сценариев запуска системы.

Пользователь TrustedUser может владеть файлами баз данных и псевдонимов. Ему разрешается запускать демон и перестраивать файл aliases. Этот пользователь необходим для поддержки графических надстроек к программе sendmail, которые вынуждены предоставлять ограниченный административный контроль определенным пользователям. Обязательно контролируйте учетную запись, соответствующую пользователю TrustedUser, поскольку через нее можно легко получить права суперпользователя. Пользователь TrustedUser не соответствует классу TRUSTED_USERS, который определяет, кто имеет право менять заголовок “From” сообщений[32].

Пользователю RunAsUser соответствует значение UID, которое присваивается программе sendmail при открытии сокета, подключенного к порту 25. Привилегированные порты (их номера не превышают 1024) могут открываться только суперпользователем, следовательно, программа sendmail первоначально должна работать от имени пользователя root. Тем не менее, по завершении указанной операции программа может взять себе другое значение UID. Это уменьшает вероятность возможных злоупотреблений, если программа sendmail запускается обманным путем. Не применяйте средство RunAsUser на компьютерах, где есть другие пользовательские учетные записи и службы; оно предназначено для брандмауэров и бастионных хостов.

По умолчанию программа продолжает выполняться от имени пользователя root. Если же учетная запись RunAsUser не эквивалентна суперпользователю, придется многое изменить. Пользователю RunAsUser должна принадлежать очередь почтовых сообщений, он должен иметь возможность читать все подключаемые файлы и файлы баз данных, запускать нужные программы и т.д. На поиск всех файлов и каталогов, владелец которых должен быть изменен, может уйти несколько часов.

[32] Средство TRUSTED_USERS обычно применяется для поддержки программ, работающих со списками рассылки. Например, если используется программа Majordomo, нужно добавить пользователя majordom в класс TRUSTED_USERS. По умолчанию членами этого класса являются пользователи root и daemon.

Права доступа

Для безопасности программы sendmail большое значение имеют права доступа к определенным файлам и каталогам. Установки, приведенные в табл. 18.14, считаются безопасными.

Таблица 18.14. Атрибуты каталогов программы sendmail

Каталог	Владелец	Режим	Назначение/содержимое
/var/spool/clientmqueue	smmsp	770	Почтовая очередь для начальной подачи почты[a]
/var/spool/mqueue	RunAsUser	700	Очередь почтовых сообщений
/, /var, /var/spool	root	755	Родительские каталоги подкаталога mqueue
/etc/mail/*	TrustedUser	644	Таблицы, конфигурационный файл, файлы псевдонимов
/etc/mail	TrustedUser	755	Родительский каталог для разных файлов
/etc	root	755	Путь к каталогу mail

[a] В версии 8.12 и выше.

Программа sendmail не будет читать файлы с "ослабленными" правами доступа (например, файлы, открытые для записи, а также файлы, находящиеся в каталогах, которые открыты для записи). Такая предосторожность вызвана тем, что многие операционные системы позволяют пользователям отдавать свои файлы в "чужие руки" с помощью команды chown (в основном это относится к системам, производным от System V).

В Linux имеется вполне нормальная версия команды chown, которая не разрешает передачу полномочий. Но в ядре есть специальная макроконстанта (CAP_CHOWN) для включения семантики System V. После ее изменения требуется перекомпилировать ядро. Мы не советуем так поступать.

Программа sendmail очень тщательно проверяет путь к любому файлу псевдонимов и файлу .forward, что иногда сказывается на управлении списками рассылки программы Majordomo. Если список рассылки находится, например, в каталоге /usr/local, то весь путь должен быть надежным; в частности, ни один его компонент не должен иметь право записи для группы. Это усложняет владельцу списка работу с файлом псевдонимов. Чтобы проверить текущее положение дел, запустите команду

```
# sendmail -v -bi
```

Флаг -bi заставляет программу проинициализировать базу данных псевдонимов и выдать предупреждение, если права доступа не подходят.

Программа sendmail больше не читает файл .forward, если количество ссылок на него больше единицы, а путь к каталогу небезопасен (права доступа ослаблены). На этом не так давно попалась Эви Немет, когда ее файл .forward, представляющий собой жесткую ссылку либо на файл .forward.to.boulder, либо на файл .forward.to.sandiego, перестал поддерживать пересылку почты от небольшого хоста, с которого ей иногда приходили сообщения. Прошли месяцы, прежде чем Эви поняла, что это происходит по ее вине и фраза "Я никогда не получала от вас почту" не является оправданием.

Отключить многие ограничения на доступ к файлам можно с помощью опции DontBlameSendmail. Но не забывайте о безопасности.

Безопасная пересылка почты в файлы и программы

В качестве агента программной доставки мы рекомендуем применять утилиту smrsh, а не /bin/sh; агентом локальной доставки лучше назначить утилиту mail.local, а не /bin/mail. Обе утилиты входят в дистрибутив sendmail. Для их активизации необходимо добавить в mc-файл следующие директивы:

```
FEATURE(`smrsh', `каталог')
FEATURE(`local_lmtp', `каталог')
```

Если каталоги не указаны, по умолчанию принимается каталог /usr/libexec. Можно воспользоваться опцией confEBINDIR, чтобы изменить стандартный каталог исполняемых файлов. В Red Hat (имеется в виду стандартный вариант инсталляции) вообще нет утилиты mail.local. В SUSE обе они находятся в каталоге /usr/bin/sendmail.d/bin, а в Debian и Ubuntu — в каталоге /usr/lib/sm.bin.

Утилита smrsh запускает программы, находящиеся только в одном каталоге (по умолчанию — /usr/adm/sm.bin), игнорируя заданные пользователями имена каталогов. Red Hat и Fedora инсталлируют исполняемый файл smrsh в каталог /usr/sbin, SUSE помещает его в каталог /usr/lib/sendmail.d/bin, а Debian и Ubuntu — в /usr/lib/sm.bin. Утилита smrsh игнорирует пути, определенные пользователями, и старается отыскать любые запрошенные команды в известном ей безопасном каталоге. Она также блокирует некоторые метасимволы интерпретатора команд, в частности "<" — операцию переадресации входного потока. В каталоге sm.bin допускается наличие символических ссылок, поэтому создавать копии программ не потребуется[33].

Приведем несколько примеров команд вместе с их возможными интерпретациями утилитой smrsh:

```
vacation eric                    # выполняется команда
                                 # /usr/adm/sm.bin/vacation eric
cat /etc/passwd                  # отклоняется, команды cat нет
                                 # в каталоге sm.bin
vacation eric < /etc/passwd      # отклоняется, оператор < не разрешен
```

Опция SafeFileEnvironment программы sendmail контролирует, куда можно записывать сообщения, если в файле aliases или .forward задано перенаправление почты в файл. Это опция заставляет программу осуществить системный вызов chroot, вследствие чего корневым каталогом файловой системы станет не /, а /safe или любой другой указанный в опции каталог. Например, если в файле псевдонимов задается перенаправление почты в файл /etc/passwd, то на самом деле сообщения будут записываться в файл /safe/etc/passwd.

Опция SafeFileEnvironment защищает также файлы устройств, каталоги и другие специальные файлы, позволяя записывать почту только в обычные файлы. Это полезно не только в плане улучшения безопасности, но и с точки зрения борьбы с ошибками пользователей. В некоторых системах значение этой опции задается равным /home, что

[33] Не помещайте в каталог sm.bin программы наподобие procmail, которые способны запускать интерпретатор команд. И не используйте программу procmail в качестве локального агента доставки, так как пользователи смогут запускать любые программы из файла ~/.procmailrc. А это небезопасно.

открывает доступ к начальным каталогам пользователей, но оставляет системные файлы вне досягаемости.

Агенты доставки тоже могут запускаться в среде с измененным корневым каталогом. В настоящий момент соответствующая опция содержится в настройках агента, но в скором времени она "перекочует" в файл конфигурации `sendmail`.

Опции безопасности

В программе `sendmail` есть опции безопасности, которые определяют:

- какие сведения о системе можно узнать из внешнего мира по протоколу SMTP;
- что требуется от хоста на противоположном конце SMTP-соединения;
- могут ли пользователи просматривать или обрабатывать очередь почтовых сообщений.

В табл. 18.15 приведены текущие значения опций безопасности. Самую последнюю информацию можно узнать в файле `doc/op/op.ps` дистрибутива.

Таблица 18.15. Значения опции `confPRIVACY_FLAGS`

Значение	Интерпретация
`public`	Проверка конфиденциальности/безопасности не осуществляется
`needmailhelo`	Требуется SMTP-команда `HELO` (идентифицирует удаленный хост)
`noexpn`	Не допускается SMTP-команда `EXPN`
`novrfy`	Не допускается SMTP-команда `VRFY`
`needexpnhelo`	Не допускается раскрытие адреса (команда `EXPN`) без команды `HELO`
`needvrfyhelo`	Не допускается проверка адреса (команда `VRFY`) без команды `HELO`
`noverb`[а]	Не допускается "многословный" режим команды `EXPN`
`restrictmailq`	Только пользователи группы, которой принадлежит каталог `mqueue`, могут просматривать очередь сообщений
`restrictqrun`	Только владелец каталога `mqueue` может обрабатывать очередь сообщений
`restrictexpand`	Ограничивается объем информации, выдаваемой при наличии флагов `-bv` и `-v`[б]
`noetrn`[в]	Не допускается асинхронная обработка очереди
`authwarnings`	К сообщениям добавляется заголовок "Authentication-Warning" (это установка по умолчанию)
`noreceipts`	Запрещается выдача кодов состояния доставки
`nobodyreturn`	При выдаче кода состояния доставки не возвращается тело сообщения
`goaway`	Отменяются все статусные SMTP-запросы (`EXPN`, `VRFY` и т.д.)

[а] В "многословном" режиме команда EXPN отслеживает переадресацию в файле `.forward` и выдает дополнительную информацию о местонахождении пользовательской почты. Необходимо включать опцию `noverb` или, еще лучше, `noexpn` на любом компьютере, имеющем выход во внешний мир.

[б] Если только программа не выполняется от имени пользователя `root` или `TrustedUser`.

[в] ETRN — это команда протокола ESMTP, которая используется хостами с коммутируемым доступом. Она запрашивает обработку очереди только для сообщений данного хоста.

Мы рекомендуем сделать в mc-файле "консервативные" установки:

```
define(`confPRIVACY_FLAGS', ``goaway, authwarnings,
      restrictmailq, restrictqrun'')
```

По умолчанию установлена лишь опция authwarnings. Обратите внимание на сдвоенные кавычки: некоторые версии препроцессора m4 требуют этого для предотвращения ненужной интерпретации запятых в списке значений. В Red Hat и Fedora по умолчанию принята установка authwarnings, а в SUSE, Debian и Ubuntu — установки authwarnings, needmailhelo, novrfy, noexpn и noverb.

Изменение корневого каталога программы sendmail

Те, кого беспокоит влияние программы sendmail на файловую систему, могут запускать ее в среде с измененным корневым каталогом. Создайте в этом каталоге минимальную файловую систему, включив в нее файл /dev/null, важные файлы каталога /etc (passwd, group, resolv.conf, sendmail.cf, таблицы баз данных, mail/*), необходимые программе sendmail совместно используемые библиотеки, исполняемый файл sendmail, каталог очереди почтовых сообщений и журнальные файлы.

Команда для запуска программы sendmail будет выглядеть так:

```
# chroot /jail /usr/sbin/sendmail -bd -q30m
```

Отражение атак типа "отказ в обслуживании"

Атаки типа "отказ в обслуживании" невозможно предотвратить, потому что нельзя заранее предсказать, является ли сообщение оружием атаки или нет. Злоумышленники могут поступать по-разному, например переполнять SMTP-порт ложными запросами на подключение, забивать разделы диска гигантскими сообщениями, засорять выходные соединения, бомбардировать систему почтовыми сообщениями и т.д. В программе sendmail имеются конфигурационные параметры, позволяющие ограничить влияние этих атак на систему, но в результате может пострадать и легитимная почта. Определенную помощь может оказать и библиотека функций фильтрации почты.

Опция MaxDaemonChildren ограничивает число процессов sendmail. Она не позволяет программе захватить все вычислительные ресурсы системы, но зато дает возможность хакерам очень легко "завалить" службу SMTP. Опция MaxMessageSize защищает каталог очереди от переполнения, но если задать значение опции слишком маленьким, "жертвами" могут оказаться обычные сообщения. Следует уведомить пользователей о существующем пределе на размер сообщения, чтобы они не удивлялись, если письмо вдруг вернется обратно. Этот предел обычно устанавливается достаточно большим.

Опция ConnectionRateThrottle задает предельно допустимое число соединений в секунду. Она может привести к небольшому замедлению работы программы sendmail. Наконец, опция MaxRcptsPerMessage определяет максимальное количество получателей сообщения.

Программа sendmail всегда умела отказываться от приема сообщений (опция REFUSE_LA) и ставить сообщения в очередь (опция QUEUE_LA) в зависимости от уровня загруженности системы. В версии 8.12 появилась дополнительная опция DELAY_LA, позволяющая продолжить обработку почты, но с меньшей скоростью. Об этом будет рассказываться в конце раздела 18.14.

Несмотря на все меры предосторожности, трудно предотвратить ситуацию, когда кто-то начинает бомбардировать систему письмами. Это может стать очень неприятной проблемой.

В университете штата Колорадо для каждого студента (всего их около 25 000) создается почтовая учетная запись, а стандартным почтовым клиентом является утилита `pine`. Несколько лет назад один студент устроился на работу в местный компьютерный магазин, и хозяин попросил студента предоставить ему копию файла паролей. Затем компания разослала рекламное сообщение всем пользователям, упомянутым в файле паролей, пакетами по 1000 получателей за раз. В результате заголовок "To" получился очень длинным.

Утилита `pine` была скомпилирована так, чтобы по умолчанию отвечать не только отправителю, но и всем получателям. Многие студенты ответили на рекламу письмами с вопросами вроде "Зачем вы прислали мне эту чушь?", и, естественно, письма оказались разосланы всем, кто был упомянут в заголовке "To". В результате сервер был полностью выведен из строя. Он не мог обрабатывать не только почту, но и любые другие запросы. Программа `sendmail` захватила все ресурсы процессора, очередь почтовых сообщений стала огромной, и все выполняемые задания зависли. Пришлось отключать компьютеры от сети, а затем вручную удалять сообщения из почтового буфера каждого пользователя.

Фальсификации

Подделать почтовое сообщение раньше было, что называется, "раз плюнуть". Раньше любой пользователь мог сфальсифицировать письмо, в результате чего оно выглядело так, будто было отправлено из чьего-то другого домена. В программу `sendmail` версии 8.10 и выше входят средства SMTP-аутентификации, позволяющие проверять подлинность хоста-отправителя. Аутентификацию нужно включать с помощью опции `AuthMechanisms`. К сожалению, программа `sendmail` поддерживает аутентификацию не между отправителем и получателем, а между смежными серверами. Если сообщение обрабатывается несколькими серверами, аутентификация не гарантирует подлинности сообщения.

В почтовых сообщениях можно представиться кем угодно. Будьте осторожны, если электронная почта является средством аутентификации для таких вещей, как ключи шифрования, карточки доступа и электронные деньги. Необходимо предупредить пользователей, чтобы в случае появления подозрительной почты от якобы уполномоченного лица они проверяли достоверность таких сообщений. Это вдвойне необходимо, если в сообщении содержится просьба предоставить неоправданные льготы какому-то лицу, не имеющему к ним никакого отношения. Например, очень подозрительно, если письмо содержит просьбу выдать студенту ключ шифрования.

При установленной опции `authwarnings` программа `sendmail` добавляет заголовок "Authentication-Warning" к любому исходящему сообщению, которое выглядит поддельным. Однако многие пользовательские почтовые агенты по умолчанию скрывают этот заголовок.

Если поддельная почта поступает с компьютера, находящегося под вашим административным контролем, можно попытаться отловить нарушителя. Демон `identd` позволяет узнать реальное регистрационное имя отправителя письма. Программа `sendmail` посылает обратный запрос к хосту-отправителю, чтобы работающий на нем демон `identd` сообщил ей имя пользователя, пославшего сообщение. Если демон на удаленном хосте не работает, то ничего выяснить не удастся. Когда удаленный компьютер является однопользовательской рабочей станцией, ее владелец может настроить демон `identd` так,

чтобы тот возвращал заведомо неправильный ответ. Если же удаленный компьютер — это большая многопользовательская система (такая, например, как во многих университетских вычислительных центрах), то демон `identd` возвращает реальное регистрационное имя пользователя, которое программа `sendmail` включает в заголовок сообщения.

Во многих организациях демон `identd` не используется: он блокируется брандмауэром. Применять демон имеет смысл только внутри организации, так как компьютерам, находящиеся под посторонним административным контролем, нельзя доверять. В то же время в крупных организациях он влияет на производительность программы `sendmail`.

Несколько лет назад, когда мы впервые начали экспериментировать с демоном `identd`, один наш студент решил подшутить над своими коллегами. Он начал посылать им письма от имени руководителя проекта, в которых говорил, что они плохо выполняют свою работу и должны работать лучше. Но однажды он допустил синтаксическую ошибку, и письмо вернулось руководителю. Программа `sendmail` благодаря протоколу IDENT сама сообщила нам имя нарушителя, включив в сообщение такую строку:

```
The original message was received at Wed, 9 Mar 1994 14:51 -0700
   from студент@benji.Colorado.EDU [128.138.126.10]
```

А в заголовке было сказано нечто иное:

```
From: руководитель@cs.Colorado.EDU
```

Мораль: занимаясь хакерством, хотя бы не допускайте синтаксических ошибок! В соответствии с нашими правилами, учетная запись студента была отключена до конца семестра, против чего он не возражал. Он не справлялся с курсовым проектом, и его коллегам приходилось часто выполнять работу за него.

Безопасность сообщений

Истинную безопасность электронной почты невозможно соблюсти, если не применять внешнюю систему шифрования, такую как PGP или S/MIME. По умолчанию вся почта посылается в незашифрованном виде. Двустороннее шифрование требует поддержки со стороны пользовательских агентов.

📖 О программе PGP рассказывается в разделе 20.11.

Системы S/MIME и PGP описаны в документах RFC, причем первая приобретает статус стандарта. Мы же предпочитаем PGP, так как эта программа широко доступна и написана великолепным шифровальщиком Филом Циммерманом (Phil Zimmermann), которому мы доверяем. Эти системы заложили основу для обеспечения конфиденциальности, аутентификации, контроля целостности и строгой идентификации отправителя сообщения. Возможен и анализ трафика, так как заголовки и конверт посылаются в текстовом виде.

Объясните пользователям, что они сами должны выполнять шифрование, если хотят обеспечить конфиденциальность своих сообщений.

SASL: простой протокол аутентификации и защиты

Программа `sendmail` версии 8.10 и выше поддерживает систему SMTP-аутентификации, которая определена в документе RFC2554. Она основана на технологии SASL (Simple Authentication and Security Layer — простой протокол аутентификации и защиты), представляющей собой систему совместного использования секретного ключа между двумя хостами. В ней нужно явно задавать пары серверов, осуществляющих взаимную аутентификацию.

SASL — это базовый механизм аутентификации, который может быть встроен в ряд других протоколов. Пока что его используют программа `sendmail`, демон `imapd` компании Cyrus, почтовые программы Outlook, Mulberry и некоторые версии Eudora. В протоколе SASL реализованы две основные концепции: идентификатор авторизации и идентификатор аутентификации. Эти идентификаторы могут быть связаны с правами доступа к файлам, паролями учетных записей, мандатами Kerberos и т.д. Библиотека функций SASL состоит из двух частей: модуля аутентификации и модуля шифрования. Библиотеку Cyrus SASL можно найти по адресу `ftp://ftp.andrew.cmu.edu/pub/cyrus-mail`.

Еще одна система аутентификации и шифрования — TLS — определена в документе RFC2487. Она реализована в программе `sendmail` в виде расширения протокола SMTP под названием STARTTLS. Можно даже одновременно использовать протоколы SASL и TLS.

Протокол TLS труднее настроить, так как требуется поддержка со стороны центра сертификации. Можно заплатить компании VeriSign, которая выдаст вам необходимые сертификаты (подписанные открытые ключи, идентифицирующие их предъявителя), или организовать собственный центр сертификации. В процессе ретрансляции почты и приема запросов от других хостов аутентификация осуществляется не на основе имени хоста или его IP-адреса, а путем обмена ключами. Записи вида

```
TLS_Srv:secure.example.com ENCR:112
TLS_Clt:laptop.example.com PERM+VERIFY:112
```

в базе доступа указывают на то, что используется механизм STARTTLS. Почта, направляемая в домен secure.example.com, должна шифроваться как минимум 112-битовыми ключами. Почта, поступающая от хоста laptop.example.com, будет приниматься лишь в том случае, если клиент пройдет процедуру аутентификации.

Грегори Шапиро (Gregory Shapiro), тоже из компании Sendmail, Inc., составил отличные учебники по безопасности программы `sendmail`, доступные по адресу `www.sendmail.org/~gshapiro`.

18.13. Производительность программы sendmail

Программа `sendmail` имеет ряд конфигурационных опций, предназначенных для повышения производительности. Мы упоминали их в разных частях главы, а теперь попытаемся более подробно описать наиболее важные из них. Эти опции и средства необходимо учитывать при развертывании крупномасштабной почтовой системы. Но, конечно, если пропускная способность системы должна составлять порядка миллиона сообщений в час, имеет смысл приобрести коммерческую версию программы `sendmail`, предлагаемую компанией Sendmail, Inc.

Режимы доставки

Программа `sendmail` имеет четыре режима доставки: фоновый, интерактивный, с постановкой в очередь и отложенный. Каждый из них представляет собой определенный компромисс между временем задержки и пропускной способностью. В фоновом режиме сообщение доставляется немедленно, но для этого программа должна породить новый процесс. В интерактивном режиме тоже поддерживается немедленная доставка, но ее осуществляет тот же процесс, поэтому удаленный хост вынужден дожидаться результатов. В режиме очереди поступившее сообщение помещается в буфер, откуда позднее его извлекает обработчик очереди. Режим отложенной доставки аналогичен предыдущему,

но в очередь заносятся также все операции поиска в таблицах, в DNS, в файлах aliases и .forward. Интерактивный режим применяется редко. Фоновый режим обеспечивает меньшее время задержки, а в режиме отложенной доставки и режиме очереди выше пропускная способность. Режим доставки задается с помощью опции confDELIVERY_MODE. По умолчанию принят фоновый режим.

Группы очередей и разбивка конвертов

Группы очередей появились в программе sendmail версии 8.12. Теперь стало возможным создавать несколько очередей для исходящей почты и управлять атрибутами каждой группы по отдельности. Группа очередей может состоять из одного или нескольких каталогов. Например, если Linux-система используется в качестве почтового концентратора на хосте провайдера, то можно определить группу очередей для коммутируемых пользователей и дать каждому пользователю право инициализировать обработку очереди (с помощью SMTP-команды ETRN) при загрузке почты. С понятием групп очередей тесно связан механизм разбивки конвертов, позволяющий распределить конверт с множеством получателей между несколькими группами очередей. Это помогает решить проблему, связанную с нахождением слишком большого числа файлов в одном почтовом каталоге[34].

Когда сообщение поступает в почтовую систему, оно закрепляется за одной или несколькими группами очередей. Для каждого получателя сообщения группа определяется независимо. Происходит перезапись адресов конверта в соответствии с установками соответствующей группы. Если используется несколько каталогов очередей, сообщения произвольным образом распределяются между очередями группы. Конверт, число получателей которого превышает предел получателей для данной группы, разбивается на несколько меньших конвертов, соответствующих параметрам группы.

Группы очередей объявляются посредством директив mc-файла, но в действительности они конфигурируются в блоке LOCAL_RULESETS низкоуровневого файла конфигурации, который в книге не описывается. Приводятся лишь примеры, которые помогут читателям сориентироваться в этой теме.

В табл. 18.16 перечислены атрибуты групп очередей. Только первую букву названия атрибута необходимо указывать в определении группы.

Таблица 18.16. Атрибуты групп очередей

Атрибут	Назначение
Flags	В основном предназначен для будущих расширений программы; флаг f задает использование нескольких обработчиков очереди
Nice	Приоритет данной группы очередей
Interval	Интервал между запусками обработчика очереди
Path	Путь к каталогу очереди, связанному с данной группой (обязателен)
Runners	Число процессов sendmail, которые должны одновременно запускаться для обработки группы очередей
recipients	Максимальное число получателей конверта

[34] Если вы используете файловую систему ext3 с ядром 2.6, или более свежей его версией, то благодаря индексам каталогов можно еще больше снизить влияние на производительность, оказываемое крупными каталогами. Чтобы использовать этот вариант, можно изменить существующие файловые системы с помощью команды tune2fs -O dir_index.

Приведем пример, в котором задаются группы очередей для локальной почты, для почты хоста aol.com, для почты других удаленных хостов, а также стандартная очередь для остальной почты. Следующие строки должны находиться в основной части mc-файла:

```
dnl ##### -- очереди
QUEUE_GROUP(`local', `P=/var/spool/mqueue/local')
QUEUE_GROUP(`aol', `P=/var/spool/mqueue/aol, F=f, r=100')
QUEUE_GROUP(`remote', `P=/var/spool/mqueue/remote, F=f')
```

А эти строки располагаются в конце mc-файла:

```
LOCAL_RULESETS
Squeuegroup
R<$+>                $1
R$*@aol.com          $# aol
R$*@mydomain.com     $# local
R$*@$*               $# remote
R$*                  $# mqueue
```

В этом примере на группу очередей хоста aol.com наложено ограничение: каждое сообщение должно иметь не более 100 получателей. Если исходящее сообщение адресовано 10 000 получателей, причем 1234 из них находятся на хосте aol.com, то при разбивке конверта в группу очередей aol будут занесены 13 сообщений, из которых 12 имеют 100 получателей и одно — 34 получателя.

Чтобы еще больше ускорить обработку сообщений, включите режим быстрой разбивки конвертов, в котором поиск записей MX осуществляется по окончании сортировки:

```
define(`confFAST_SPLIT', `1')
```

Обработчики очередей

Программа sendmail порождает копии самой себя при транспортировке сообщений. Можно задавать, сколько копий программы должно выполняться в любой момент времени и даже сколько из них должно быть связано с каждой группой очередей. Это позволяет системным администраторам перераспределять нагрузку между программой sendmail и операционной системой на крупных почтовых концентраторах.

Количество демонов sendmail, обрабатывающих каждую очередь, определяется тремя опциями:

- MAX_DAEMON_CHILDREN — задает общее число демонов sendmail, которые могут выполняться одновременно, включая те, что обрабатывают очередь, и те, что принимают входящую почту;
- MAX_QUEUE_CHILDREN — задает общее число одновременно выполняющихся обработчиков очереди;
- MAX_RUNNERS_PER_QUEUE — задает стандартный предел числа обработчиков одной очереди, если в определении группы отсутствует параметр Runners= (или R=).

Если введенные значения могут вызвать конфликт (например, общее число обработчиков равно 50, но при этом для локальной очереди задано 10 обработчиков, для домена mydomain.com — 30, а для домена aol.com — 50), то программа sendmail разбивает очереди на рабочие группы и обслуживает их по круговому принципу. В данном примере локальная очередь и очередь домена mydomain.com составят одну рабочую группу, а очередь домена aol.com — вторую. Если же значения явно конфликтуют (общий предел

равен 50, но для домена aol.com — 100), то максимальное число обработчиков определяется значением опции MAX_QUEUE_CHILDREN.

Контроль средней загруженности

Программа sendmail всегда умела отказываться от приема сообщений или помещать сообщения в очередь, когда показатель средней загруженности системы становится очень высоким. К сожалению, замер средней загруженности осуществляется раз в минуту, что не позволяет с достаточной степенью точности контролировать потребление программой sendmail ресурсов системы. Новая опция DELAY_LA позволяет задать показатель средней загруженности, при котором программа должна замедлить свою работу: она будет делать секундную паузу между выполнением SMTP-команд текущего соединения и приемом нового запроса на подключение. По умолчанию опция равна нулю, т.е. режим отключен.

Обработка недоставленных сообщений

Недоставленные сообщения, остающиеся в почтовой очереди, способны существенно снизить производительность почтового сервера. У программы sendmail есть ряд средств для борьбы с такими сообщениями. Наиболее эффективным из них является опция FALLBACK_MX, задающая пересылку сообщений на другой компьютер, если их не удалось доставить с первой попытки. Эта опция позволяет главному серверу заниматься доставкой почты с правильными адресами, перепоручив "проблемную" почту резервному серверу. Вторая опция, HOST_STATUS_DIRECTORY, позволяет сохранять информацию о состоянии удаленных хостов между запусками обработчиков очереди.

Опция FALLBACK_MX позволяет существенно повысить производительность хоста с большими списками рассылки, где неизбежно попадаются временно или постоянно недоступные адреса. Значением опции должен быть адрес компьютера, обрабатывающего отложенную почту. Например, команда

```
define(`confFALLBACK_MX', `mailbackup.artust.com')
```

задает пересылку всех сообщений, которые не удалось доставить с первого раза, на центральный сервер mailbackup.artust.com для последующей обработки. В версии 8.12 разрешается иметь несколько резервных серверов, если указанному хосту соответствует несколько записей MX в базе данных DNS.

На резервном сервере можно воспользоваться опцией HOST_STATUS_DIRECTORY для решения проблемы повторных сбоев. Эта опция предписывает программе sendmail хранить статусный файл для каждого компьютера, на который посылается почта. На основании этой информации определяются приоритеты просмотра хостов при очередной обработке очереди. Это позволяет реализовывать схему отрицательного кэширования и хранить статусную информацию в промежутках между запусками обработчиков очередей. Результатом становится повышение производительности серверов, обрабатывающих списки рассылки с большим числом неправильных адресов, хотя это и затратно с точки зрения файлового ввода-вывода.

В следующем примере для хранения статусной информации выделяется каталог /var/spool/mqueue/.hoststat (его нужно предварительно создать):

```
define(`confHOST_STATUS_DIRECTORY', `/var/spool/mqueue/.hoststat')
```

Если указать относительный путь к каталогу .hoststat, то он будет создан в каталоге очереди. Программа sendmail формирует собственную иерархию подкаталогов, основываясь на имени хоста-получателя.

Например, если почту по адресу evi@anchor.cs.colorado.edu не удается доставить, то статусная информация будет занесена в файл anchor каталога /var/spool/mqueue/.hoststat/edu./colorado./cs./, так как наиболее приоритетная запись MX хоста anchor ссылается на него самого. Если бы записи MX задавали перенаправление почты хоста anchor на хост foo, то имя файла было бы foo, а не anchor.

Третья возможность повышения производительности связана с заданием минимального времени пребывания сообщения в очереди. Сообщение, которое не удалось доставить с первого раза, помещается в очередь и остается там до тех пор, пока не будет осуществлена повторная попытка. Обычно это делается в сочетании с флагами командной строки, задающими более частую обработку очереди (например, -q5m). Если обработчик очереди "зависает" на каком-то сообщении, то через 5 минут запускается другой. Вся очередь обрабатывается блоками, формируемыми в зависимости от того, какие сообщения превысили минимальное время пребывания в очереди. Таким образом, запуск программы sendmail с флагами -bd -q5m и включение опции

```
define(`confMIN-QUEUE_AGE', `27m')
```

в конфигурационном файле позволяют организовать более оперативную обработку почты.

Настройка ядра

Если Linux-систему планируется использовать в качестве крупного почтового сервера, придется модифицировать ряд параметров сетевой конфигурации ядра, а возможно, даже скомпилировать новое ядро (это зависит от конфигурации оборудования и ожидаемой нагрузки). Удалите все ненужные драйверы, чтобы ядро было оптимизировано для обработки почты.

В ядре должна быть включена поддержка многопроцессорной обработки, если компьютер оснащен несколькими процессорами (мы понимаем, что это напоминает совет "Не забывайте дышать", но в действительности далеко не все пользователи умеют правильно настраивать ядро).

Для сброса параметров стека сетевых протоколов достаточно воспользоваться командой echo, перенаправив ее выходной поток в соответствующую переменную файловой системы /proc. Подробнее эта процедура описывалась в главе 12. В табл. 18.17 перечислены стандартные и рекомендуемые значения параметров, которые потребуется изменить на крупном почтовом сервере. Лучше всего занести необходимые команды echo в сценарии запуска системы.

Таблица 18.17. Параметры ядра, которые требуется менять на крупных почтовых серверах

Переменная (относительно каталога /proc/sys)	По умолчанию	Рекомендуется
net/ipv4/tcp_fin_timeout	180	30
net/ipv4/tcp_keepalive_time	7200	1800
net/core/netdev_max_backlog	300	1024
fs/file_max	4096	16384
fs/inode_max	16384	65536

Например, с помощью команды

```
echo 30 > /proc/sys/net/ipv4/tcp_fin_timeout
```

можно уменьшить продолжительность тайм-аута запросов FIN протокола TCP.

18.14. Сбор статистических данных, тестирование и отладка

Программа `sendmail` может собирать статистические данные о количестве и размере обрабатываемых сообщений. Эти данные группируются по агентам доставки и отображаются командой `mailstats`. Опция `STATUS_FILE` в файле каталога `ostype` задает имя файла, в котором должны сохраняться статистические данные. Если этот файл существует, функция сбора учетной информации включается автоматически.

По умолчанию файл статистики находится в каталоге `/etc/mail/statistics`, но некоторые поставщики называют файл `sendmail.st` и перемещают его в каталог `/var/log`. Данные накапливаются в файле с момента его создания. Поэтому, если необходима статистическая информация за определенный период, следует организовать ротацию и повторную инициализацию этого файла из демона `cron`.

Ниже приведен образец статистических данных.

```
% mailstats
Statistics from Tue Aug 1 02:13:30 2006
 M   msgsfr   bytes_from  msgsto  bytes_to  msgsrej  msgsdis Mailer
 4       12          25K      63      455K        0        0  esmtp
 7        0           0K      18       25K        0        0  relay
 8       54         472K       0        0K        0        0  local
=============================================================
 T       66         497K      81      480K        0        0
 C       66                   81                0
```

Если файл статистики открыт для чтения, то для запуска команды `mailstats` не нужно быть суперпользователем.

В примере показано шесть значений: число полученных сообщений и их суммарный объем в килобайтах (`msgsfr`, `bytes_from`), число отправленных сообщений и их суммарный объем в килобайтах (`msgsto`, `bytes_to`), число отклоненных сообщений (`msgsrej`) и число проигнорированных сообщений (`msgsdis`). В первой колонке содержится идентификатор агента доставки, а в последней колонке указано его имя. В строке `T` показано суммарное число сообщений и их объем в рамках всей почтовой системы. В строке `C` показано суммарное число соединений. Эти значения относятся как к локальной, так и к ретранслируемой почте.

Тестирование и отладка

Конфигурация, основанная на применении препроцессора `m4`, в определенной степени тестируется автоматически. Низкоуровневая отладка в основном не требуется. Единственное, что невозможно контролировать при помощи флагов отладки, — это смысловые ошибки. В процессе подготовки главы мы обнаружили несколько подобного рода ошибок в конфигурационных файлах. Иногда, к примеру, вызывалось средство без сопутствующего ему макроса (в частности, активизировалось средство `masquerade_envelope`, но не включался режим маскирования адресов с помощью мак-

роса MASQUERADE_AS). Или конфигурация программы sendmail не соответствовала настройкам брандмауэра, который определял, какую почту разрешается пропускать.

Нельзя проектировать почтовую систему саму по себе. Ее необходимо согласовать с имеющимися записями MX базы данных DNS и настройками брандмауэра.

Программа sendmail располагает богатым комплектом отладочных средств. Отладочные флаги представляют собой не просто булевы значения и даже не целые числа, а двухмерные величины *x*.*y*, где *x* определяет тему, а *y* — объем выдаваемой информации (уровень отладки). Уровень 0 соответствует отсутствию отладки, а уровень 127 — максимально подробному отчету. Темы могут иметь номера от 0 до 99 (в настоящее время определено около восьмидесяти тем).

В файле sendmail/TRACEFLAGS дистрибутива содержится список поддерживаемых значений, а также перечень файлов и функций, в которых они используются. Вся поддержка отладочных флагов реализована в низкоуровневом файле конфигурации.

Если программа sendmail вызывается с флагом -d*x*.*y*, то отладочная информация выдается на экран (стандартный поток ошибок). Некоторые важные значения *x* и рекомендуемые значения *y* приведены в табл. 18.18. Будьте осторожны, если включается режим отладки программы sendmail, работающей в режиме демона (-bd), поскольку отладочная информация может перемешаться с сообщениями SMTP-диалога и вызвать сбои при взаимодействии с удаленными хостами.

Таблица 18.18. Параметры отладки

Номер темы	Описание и предлагаемые уровни отладки
0	Отображение флагов компиляции, а также информации об операционной системе (попробуйте значения *y*, равные 1 или 10)
8	Выдача информации о преобразовании имен в DNS (попробуйте значение *y*, равное 8)
11	Трассировка процесса доставки (отображаются вызовы агентов доставки)
12	Выдача информации о преобразовании локальных имен в удаленные
17	Выдача списка записей MX
21	Трассировка правил подстановки (попробуйте значения *y*, равные 2 или 12)
27	Отображение данных о псевдонимах и перенаправлениях (попробуйте значение *y*, равное 4)
44	Отображение сведений о неудачных попытках открытия файлов (попробуйте значение *y*, равное 4)
60	Выдача информации о поиске в базах данных

Джин Ким (Gene Kim) и Роб Колстад (Rob Kolstad) написали Perl-сценарий checksendmail, который вызывает программу sendmail в режиме проверки адресов, передавая ей файл тестовых адресов (создается пользователем). Этот сценарий сравнивает полученные результаты с ожидаемыми. Он позволяет сформировать тестовый набор типичных адресов и впоследствии проверять новые версии конфигурационных файлов на предмет того, не нарушилась ли случайно работоспособность программы.

📖 Сценарий checksendmail можно получить на Web-сайте www.harker.com.

Доставка в протокольном режиме

Многие пользовательские агенты, вызывающие программу sendmail из командной строки, поддерживают флаг -v, при наличии которого программа выдает информацию обо всех действиях, выполняемых в процессе доставки сообщения. Следующий пример получен с помощью почтового агента mail. Текст, набранный полужирным шрифтом, вводится пользователем, а все остальное — это результаты работы программы sendmail.

```
% mail -v trent@toadranch.com
Subject: just testing, please ignore
hi
.
Cc:
trent@toadranch.com... Connecting to coyote.toadranch.com. via esmtp...
220 coyote.toadranch.com ESMTP Sendmail 8.11.0/8.11.0; Tue,
  7 Aug 2001 20: 08:51 -0600
>>> EHLO anchor.cs.colorado.edu
250-coyote.toadranch.com Hello anchor.cs.colorado.edu
 [128.138.242.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-EXPN
250-VERB
250-8BITMIME
250-SIZE
250-DSN
250-ONEX
250-ETRN
250-XUSR
250-AUTH DIGEST-MD5 CRAM-MD5
250 HELP
>>> MAIL From:<evi@anchor.cs.colorado.edu> SIZE=65
250 2.1.0 <evi@anchor.cs.colorado.edu>... Sender ok
>>> RCPT To:<trent@toadranch.com>
250 2.1.5 <trent@toadranch.com>... Recipient ok
>>> DATA
354 Enter mail, end with "." on a line by itself
>>> .
250 2.0.0 f7828pi03229 Message accepted for delivery
trent@toadranch.com... Sent (f7828pi03229 Message accepted for delivery)
Closing connection to coyote.toadranch.com.
>>> QUIT
221 2.0.0 coyote.toadranch.com closing connection
```

Программа sendmail на компьютере anchor связалась с программой sendmail на хосте toadranch.com. Оба компьютера в процессе обмена сообщениями взаимодействовали по протоколу ESMTP.

Обмен данными по протоколу SMTP

В процессе отладки почтовой системы можно непосредственно выполнять команды протокола SMTP. Чтобы начать SMTP-сеанс, установите с помощью программы telnet соединение с TCP-портом 25 или воспользуйтесь командой sendmail -bs. Порт 25 контролируется программой sendmail по умолчанию, когда она работает в режиме демона

(-bd). Если же программа работает в качестве агента подачи почты, то она контролирует порт 587. Наиболее важные SMTP-команды приведены в табл. 18.19.

Таблица 18.18. Команды протокола SMTP

Команда	Функция
HELO *имя_компьютера*	Идентифицирует компьютер, устанавливающий соединение по протоколу SMTP
EHLO *имя_компьютера*	Идентифицирует компьютер, устанавливающий соединение по протоколу ESMTP
MAIL From: *обратный_путь*	Инициирует почтовую операцию (отправитель конверта)
RCPT To: *прямой_путь*[a]	Идентифицирует получателей конверта
VRFY адрес	Проверяет действительность адреса (т.е. можно ли по нему доставить почту)
EXPN адрес	Раскрывает псевдонимы и соответствия, заданные в файле .forward
DATA	Начинает тело сообщения[б]
QUIT	Завершает обмен и закрывает соединение
RSET	Сбрасывает состояние соединения
HELP	Выводит список SMTP-команд

[a] Для одного сообщения может быть несколько команд RCPT.

[б] Тело завершается строкой, состоящей из одной точки.

В протоколе SMTP всего 14 команд, поэтому он прост в изучении и применении. Регистр символов здесь не важен. Полная спецификация протокола приведена в документе RFC821 (см. также RFC1123).

Большинство транспортных агентов, включая sendmail, понимает оба протокола: и SMTP, и ESMTP. Исключение составляет лишь агент smap. К сожалению, многие брандмауэры, осуществляющие активную фильтрацию, не поддерживают протокол ESMTP.

В протоколе ESMTP диалог начинается командой EHLO, а не HELO. Если в ответ на эту команду противоположная сторона возвращает сообщение OK, то участники диалога обмениваются информацией о поддерживаемых расширениях и приходят к "общему знаменателю". Если же возвращается сообщение об ошибке, то осуществляется переход на протокол SMTP.

Мониторинг очередей

Для просмотра состояния сообщений, находящихся в очереди, используется команда mailq (она эквивалентна команде sendmail -bp). Сообщения ставятся в очередь во время их доставки, или когда доставка была предпринята, но не была выполнена.

Команда mailq печатает итоговые данные, доступные для чтения пользователями, о файлах в каталоге /var/spool/mqueue в любой заданный момент времени. На основе выходных данных можно узнать причину задержки доставки сообщения. Если все говорит о том, что в данный момент создается журнал недоставленных сообщений, вы можете проследить за тем, как программа sendmail пытается разрешить данную проблему.

Имейте в виду, что версия 8.12 программы sendmail предлагает две стандартные очереди: одна из них предназначена для сообщений, доставленных через порт 25, а другая — для сообщений, доставленных через порт 587 (очередь представления клиентов).

Ниже показан пример результатов выполнения команды mailq. В данном случае речь идет о трех сообщениях, ожидающих доставки:

```
$ sudo mailq
/var/spool/mqueue (3 requests)
----Q-ID----    --Size-- ----Q-Time----  --------Sender/Recipient---------
k623gYYk008732    23217    Sat Jul 1 21:42   MAILER-DAEMON
   8BITMIME    (Deferred: Connection refused by agribusinessonline.com.)
                                  <Nimtz@agribusinessonline.com>
k5ULkAHB032374    279     Fri Jun 30 15:46 <randy@atrust.com>
          (Deferred: Name server: k2wireless.com.: host name lookup fa)
                                  <relder@k2wireless.com>
k5UJDm72023576    2485    Fri Jun 30 13:13  MAILER-DAEMON
          (reply: read error from mx4.level3.com.)
                                  <lfinist@bbnplanet.com>
```

Если вам кажется, что вы владеете ситуацией лучше, чем программа sendmail, или если вы просто хотите, чтобы она немедленно попыталась заново доставить сообщения, находящиеся в очереди, можно выполнить команду sendmail -q. Если выполнить команду sendmail -q -v, то программа sendmail будет показывать результаты после каждой попытки отправки сообщения, что часто бывает полезно для отладки. Программа sendmail повторяет отправку сообщений, находящихся в очереди, через равные промежутки времени (как правило, через 30 минут).

Журнальная регистрация

Программа sendmail регистрирует сообщения об ошибках и своем состоянии через систему Syslog. Это делается от имени средства mail на уровнях от debug до crit. Все сообщения помечаются строкой “sendmail”. Изменить эту установку можно с помощью флага командной строки -L, что довольно удобно, если отлаживается одна из копий программы, в то время как остальные копии продолжают нормально работать.

📖 Подробнее о системе Syslog рассказывалось в главе 10.

Опция confLOG_LEVEL, заданная в командной строке или в файле конфигурации, определяет уровень важности, который программа sendmail примет в качестве порогового. Высокие пороговые значения соответствуют низким уровням важности и обеспечивают регистрацию большего объема информации.

Вспомните, что сообщения, регистрируемые в системе Syslog на определенном уровне, передаются также на все вышестоящие уровни. В файле /etc/syslog.conf определяется, куда попадет каждое сообщение.

В системах Red Hat и Fedora по умолчанию сообщения попадают в файл /var/log/maillog, в SUSE — в файл /var/log/mail, в Debian и Ubuntu — в файл /var/log/mail.log. Неужели разработчики не могут устранить эти бессмысленные расхождения, чтобы проще было писать универсальные сценарии?

В табл. 18.20 показано приблизительное соответствие между уровнями журнальной регистрации в программе sendmail и уровнями важности Syslog.

Таблица 18.20. Соотношение между пороговыми уровнями программы `sendmail` и уровнями важности Syslog

Порог	Уровень важности
0	Регистрация отключена
1	`alert` или `crit`
2	`crit`
3	`err` или `warning`
4	`notice`
5–11	`info`
>=12	`debug`

Существует удобная утилита `mreport`, написанная Джейсоном Армстронгом (James Armstrong) и доступная по адресу

```
ftp://ftp.riverdrums.com/pub/mreport
```

Она суммирует информацию из журнальных файлов программы `sendmail`. Для инсталляции утилиты достаточно выполнить команды `make` и `make install`. Ниже показаны результаты, выдаваемые утилитой в Red Hat.

```
# mreport -f -i /var/log/maillog -o mreport.out
[redhat.toadranch.com] [/var/log/maillog]

* [  7]  592601  herb@yoherb.com                trent@toadranch.com
* [  8]  505797  SNYDERGA@simon.rochester.edu   trent@toadranch.com
  [  1]  179386  steph@toadranch.com            bennettr@ci.boulder.co.us
  [  1]   65086  herb@yoherb.com                ned@xor.co
  [  7]   19029  evi@anchor.cs.colorado.edu     trent@toadranch.com
  [ 11]   17677  lunch-request@moose.org        trent@toadranch.com
  [  2]   16178  trent@toadranch.com            ned@camelspit.org
  [  3]   15229  reminders@yahoo-inc.com        herb@toadranch.com
  [  2]    4653  trent@toadranch.com            garth@cs.colorado.edu
  [  2]    1816  UNKNOWN                        trent@toadranch.com

... много строк удалено ...

======================
Total Bytes          : 7876372
Number of Records    : 192
----------------
User Unknown         : 125
----------------
Host Name            : redhat.toadranch.com
Input File           : maillog
Output File          : mreport.out
First Record         : Aug 5 04:47:31
Last Record          : Aug 7 18:16:25
----------------
Time Taken           : 24317 µs
======================
```

Если журнальные файлы почтовой системы доступны лишь суперпользователю (как и должно быть), то для запуска утилиты `mreport` необходимо воспользоваться командой `sudo` или быть пользователем `root`. Флаги и опции утилиты описаны на ее `man`-странице. В данном примере флаг `-f` требует группировать и сортировать записи по адресу отправителя. Существует аналогичный флаг `-t` в отношении адреса получателя. Флаг `-i` задает входной файл, а флаг `-o` — выходной файл.

Если планируется осуществлять регулярный запуск утилиты `mreport`, сообщите об этом пользователям. Они могут посчитать это нарушением своих прав.

18.15. Почтовая система Exim

В состав Debian и Ubuntu входит не программа `sendmail`, а транспортный агент Exim. Это более простая программа с меньшим числом функциональных возможностей, зато ее проще конфигурировать в системах, обрабатывающих небольшие объемы почты и имеющих типичную структуру.

История

Программа Exim была создана в 1995 г. Филипом Хейзелом (Philip Hazel) из Кембриджского университета и распространяется на условиях общей открытой лицензии GNU (GPL). Версия 4 Exim (exim4) была выпущена в 2002 г.; в данный момент она входит в состав дистрибутивов Debian и Ubuntu. Филип Хейзел написал также книгу *Exim Smtp Mail Server: Official Guide for Release 4*, опубликованную в 2003 г. Те, кого интересуют детали конфигурирования программы, могут обратиться к этой книге и к документации, доступной на Web-сайте `www.exim.org`. Мы же лишь кратко опишем версию программы.

Программа Exim в Linux

Как и в случае `sendmail`, программа Exim обладает дополнительными командами, выполняющими определенные функции почтовой системы. Эти функции реализуются путем вызова программы `exim` со специальными флагами командной строки. Соответствующие команды и эквивалентные им флаги перечислены в табл. 18.21.

Таблица 18.21. Почтовые утилиты Exim (с эквивалентными флагами)

Команда	Флаг	Назначение
`mailq`	`-bp`	Отображает содержимое почтовой очереди
`rsmtp`	`-bS`	Организует SMTP-соединение в пакетном режиме
`rmail`	`-i`	Существует для совместимости с командой `smail`[a]
`runq`	`-q`	Осуществляет обработку очереди

[a] Принимает сообщение, которое оканчивается строкой, состоящей из одной точки.

Конфигурирование программы Exim

В программе Exim поддерживаются три типа агентов: управляющие, маршрутные и транспортные. Первые обрабатывают локальные адреса, т.е. адреса в пределах текущего домена. Вторые обрабатывают удаленные адреса, а третьи осуществляют фактическую доставку сообщений.

Программа Exim конфигурируется примерно так же, как программы `smail3` и Postfix. Записи конфигурационного файла имеют вид *ключевое_слово = значение*. Говоря в контексте версии Exim 4, система конфигурируется на основе конфигурационных файлов, находящихся в каталоге `/etc/exim4/conf.d`. После того как вы добавите или измените один из этих файлов, вы должны будете запустить сценарий `update-exim4.conf`, который создаст файл `/var/lib/exim4/config.autogenerated`.

Файл `config.autogenerated` используется программой `exim` во время выполнения.

К счастью, конфигурационные файлы программы `exim` хорошо документированы, причем каждой переменной предшествует комментарий, где указывается, для чего она нужна, каково ее стандартное значение и каковы возможные последствия изменения этого значения. В стандартных конфигурационных файлах определено около 100 переменных, но в основном они нужны для активизации всевозможных средств, имитирующих работу программы `sendmail`.

В нашей тестовой системе понадобилось установить всего четыре переменные, чтобы почтовая система заработала. Более того, две из этих переменных связаны с безопасностью и не так уж необходимы. Вот эти установки с нашими комментариями:

```
qualify_domain = имя_домена      ### по умолчанию задается имя компьютера
local_domains = localhost:имя_домена
smtp_verify = false              ### по умолчанию включена; в данном случае
                                     отключается SMTP-команда VRFY
modemask = 002                   ### значение по умолчанию 022 предполагает,
                                     что для каждого пользователя есть группа
```

В следующем разделе мы опишем некоторые особенности Exim в контексте программы `sendmail`, что позволит вам сравнить выполняемые функции двух систем и решить, нужно ли вам инсталлировать `sendmail` вместо Exim.

Что общего между программами Exim и `sendmail`

Некоторые части программы Exim запускаются с установленным битом setuid, меняя идентификатор пользователя на `root`, как это делала программа `sendmail` раньше (до появления версии 8.12). Поэтому приходится внимательно следить за появлением очередных "заплат", связанных с безопасностью. Концепции доверенных пользователей у программ Exim и `sendmail` совпадают, благодаря чему упрощается обработка списков рассылки (поддерживается перезапись заголовков "From"). В Exim можно также назначать пользователей с административными привилегиями; им разрешено инициировать обработку почтовой очереди. SMTP-команда VRFY по умолчанию разрешена.

В случае изменения конфигурационного файла следует послать программе Exim сигнал `HUP` (с помощью команды `kill`). Идентификатор процесса хранится в файле `/var/spool/exim/exim-daemon.pid`. Журнальные сообщения обычно записываются в файлы каталога `/var/log/exim`, однако могут направляться и в систему Syslog. По умолчанию программа Exim игнорирует систему Syslog, причем конфигурируется несколько

журнальных файлов, которые изначально являются пустыми, но, тем не менее, сжимаются каждый день (пустые файлы после сжатия увеличиваются в размере!).

Программа Exim позволяет перенаправлять исходящую почту на другой компьютер в режиме SMART_HOST, а также фильтровать входящую почту на уровне хостов и отдельных пользователей. Поддерживаются виртуальные домены и база повторной доставки, напоминающая статусный каталог программы sendmail, в которой хранится информация о неудачных попытках доставки сообщений. Имеется общесистемный (не на уровне отдельных пользователей) механизм фильтрации, позволяющий сканировать почту на предмет выявления вирусов, червей и прочей "нечисти".

Есть также ряд возможностей, не поддерживаемых программой sendmail. В частности, распознаются и обрабатываются циклы переадресации и циклические псевдонимы.

Наконец, с помощью команды man -k exim можно получить информацию о том, как правильно обслуживать почтовую систему.

18.16. Postfix

Postfix представляет собой еще один альтернативный вариант программы sendmail. Проект Postfix был начат, когда Витс Венема (Wietse Venema) провел свой годичный отпуск в исследовательском центре T.J. Watson Research Center, принадлежащем фирме IBM. Проект Postfix должен был решить следующие задачи: разработать политики распространения открытого исходного кода, повысить производительность, обеспечить устойчивость, гибкость и безопасность. Этот проект является непосредственным конкурентом программы qmail, разработанной Деном Бернстайном (Dan Bernstein). Все основные дистрибутивы Linux включают программу Postfix, а в версии 10.3 Mac OS X эта программа стала использоваться в качестве почтовой системы вместо sendmail.

Главными особенностями Postfix является то, что, во-первых, с ней можно начинать работать без каких-либо приготовлений (простейшие конфигурационные файлы имеют всего пару строк), а, во-вторых, для эффективной фильтрации писем она применяет регулярные выражения и библиотеку PCRE (Perl Compatible Regular Expression — совместимые с Perl регулярные выражения). Программа Postfix совместима с программой sendmail в том смысле, что файлы aliases и .forward программы Postfix имеют тот же формат и семантику, что и файлы sendmail.

Postfix "общается" по протоколу ESMTP. Она поддерживает виртуальные домены и фильтрацию спама. Postfix не использует язык подстановки адресов, как это делает sendmail; вместо этого она производит поиски в плоских файлах таких таблиц, как Berkley DB, dbm, LDAP, NIS, NetInfo, или в базах данных, таких как MySQL.

Архитектура Postfix

Комплекс Postfix состоит из нескольких небольших, совместно работающих программ, которые посылают сетевые сообщения, принимают сообщения, осуществляют локальную доставку почты, и т.п. Связь между программами осуществляется посредством сокетов доменов UNIX или алгоритмов FIFO. Ее архитектура немного отличается от архитектуры sendmail, в которой всю работу приходится выполнять одной большой программе.

Запуск и наблюдение за всеми процессами Postfix осуществляет программа master. В ее конфигурационном файле, master.cf, перечислены вспомогательные программы, а также информация о том, в каких случаях их нужно запускать. Значения, принятые по

умолчанию в этом файле, подходят как для слишком медленных, так и для слишком быстрых компьютеров (или сетей); в общем случае вам не потребуется ничего настраивать. Одним распространенным изменением является запись комментариев в программе: например, комментарий нужно поставить напротив `smtpd`, если клиент не должен вести прослушивание на порту SMTP.

Наиболее важные программы сервера, участвующие в доставке почты, показаны на рис. 18.5.

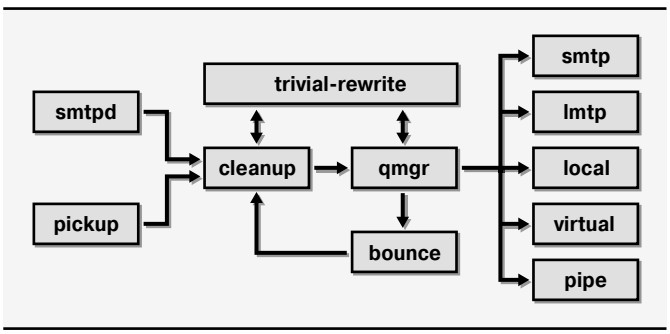


Рис. 18.5. Серверные программы Postfix

Получение почты

За прием почты на порту SMTP отвечает программа `smtpd`, которая проверяет также, имеют ли клиенты право отправлять почту, когда они пытаются осуществить ее доставку. Если электронное письмо отправляется локально посредством программы совместимости `/usr/lib/sendmail`, то файл будет записан в каталог `/var/spool/postfix/maildrop`. Этот каталог периодически сканируется программой `pickup`, которая обрабатывает любые найденные ею файлы.

Вся входящая почта обрабатывается программой `cleanup`, которая добавляет отсутствующие заголовки и переписывает адреса в соответствии с картами `canonical` и `virtual`. Прежде чем поместить письмо в очередь `incoming`, программа `cleanup` передает письмо программе `trivial-rewrite`, которая выполняет незначительные исправления в адресах, добавляя, например, почтовый домен в частично заполненный адрес.

Администратор очередей

Электронные письма, ожидающие доставки, находятся под управлением программы `qmgr`, администратора пяти очередей:

- `incoming` — поступающая почта;
- `active` — доставляемая почта;
- `deferred` — письма, доставка которых не была произведена в прошлом;
- `hold` — письма, заблокированные в очереди администратором;
- `corrupt` — письма, которые невозможно прочитать или проанализировать.

Администратор очередей обычно выбирает следующее сообщение для обработки на основе простого алгоритма FIFO (first in, fisrt out — “первым вошел, первым вышел”).

Кроме этого, он поддерживает также и сложный алгоритм преемственности, который позволяет отбирать только те сообщения, которые поступили от некоторых адресатов.

Чтобы не перегружать приемный хост, особенно после доставки писем, Postfix использует алгоритм медленного запуска, который позволяет контролировать скорость доставки корреспонденции. Отсроченные сообщения получают метку с обозначением времени следующей попытки отправки. Значения времени увеличиваются экспоненциальным образом, благодаря чему удается избежать расходования ресурсов на сообщения, которые невозможно доставить. Кэш состояния недосягаемых мест доставки позволяет избежать ненужных попыток доставки.

Отправка почты

При поддержке `trivial-rewrite` утилита `qmgr` принимает решение о том, куда должно быть отправлено сообщение. Решение о маршруте сообщения, принятое посредством `trivial-rewrite`, может быть аннулировано картой `transport`.

Доставка на дистанционные компьютеры посредством протокола SMTP осуществляется с помощью программы `smtp`. `lmtp` доставляет почту посредством протокола LMTP (Local Mail Transfer Protocol — протокол локальной пересылки почты), определенного в документе RFC2033. Протокол LMTP основан на протоколе SMTP, однако был изменен так, что для управления очередью сообщений не нужно использовать почтовый сервер. Этот почтальон особенно полезен для доставки электронной почты на серверы почтовых ящиков (почтовые серверы), такие как Cyrus IMAP.

Задача `local` заключается в локальной доставке почты. Эта программа распознает адреса в таблице `aliases` и выполняет инструкции, имеющиеся в файле `.forward` получателя. Сообщения могут направляться по другому адресу, передаваться внешней программе на обработку или сохраняться в почтовых папках пользователя.

Программа `virtual` доставляет письма в "виртуальные почтовые ящики" — почтовые ящики, которые не относятся к текущей учетной записи Linux, но которые все равно представляют действительные почтовые адреса назначения. И, наконец, программа `pipe` реализует доставку посредством внешних программ.

Защита

Postfix обеспечивает защиту на нескольких уровнях. Большинство серверных программ Postfix могут выполнять среду с измененным корневым каталогом. Они представляют собой отдельные программы, лишенные отношений между "родительский–дочерний". Ни одна их них не является setuid. Каталог, в который направляется почта, открыт для записи группе postdrop, для которой программа `postdrop` является setgid.

Примечательно, что до сих пор во всех версиях Postfix наблюдались только лишь атаки на отказ в обслуживании.

Команды и документация Postfix

Взаимодействие между пользователем и системой обработки почты регулируется несколькими утилитами командной строки:

- `sendmail`, `mailq`, `newaliases` — альтернативные варианты программы `sendmail`;
- `postfix` — запускает и прекращает работу системы обработки почты (нужны привилегии администратора);
- `postalias` — создает, модифицирует и запрашивает таблицы псевдонимов;

- `postcat` — печатает содержимое файлов очереди;
- `postconf` — отображает и редактирует главный конфигурационный файл, `main.cf`;
- `postmap` — создает, модифицирует или запрашивает таблицы преобразований;
- `postsuper` — управляет очередями почты.

Дистрибутив Postfix включает набор `man`-страниц, которые описывают все программы и их опции. Кроме того, в интерактивных документах, которые можно найти на сайте `www.postfix.org`, можно прочитать объяснение того, как конфигурируются различные аспекты Postfix и как производится управление ими. Все эти документы есть и в дистрибутиве Postfix, поэтому вы сможете найти их в каталоге `README_FILES`.

Конфигурирование Postfix

Главным конфигурационным файлом Postfix является файл `main.cf`. Он конфигурирует серверные программы. Он определяет также различные таблицы преобразований, на которые существуют ссылки в файле `main.cf`, и которые обеспечивают разные типы отображения служб.

На `man`-странице `postconf(5)` описан каждый параметр, который может быть задан в файле `main.cf`. Если просто ввести команду `man postconf`, вы попадете на `man`-страницу программы `postconf`. Чтобы попасть на `man`-страницу с описанием опций `main.cf`, нужно ввести команду `man -s 5 postconf`.

Конфигурационный язык Postfix напоминает серию комментариев и операторов присваивания интерпретатора команд Bourne. Ссылки на переменные могут производиться в определении других переменных с использованием приставки `$`. Определения переменных сохраняются по мере того, как они появляются в конфигурационном файле; они не расширяются до тех пор, пока не будут использоваться, и до этого момента в не могут производиться какие-либо замены.

Новые переменные можно создавать путем присваивания им значений. Выбирать имена следует обдуманно, чтобы не допустить возникновения конфликтов с имеющимися конфигурационными переменными.

Все конфигурационные файлы Postfix, включая таблицы преобразований, рассматривают строки, начинающиеся с пробела, как конфигурационные строки. Благодаря этому условию конфигурационные файлы легко читаются, хотя вам придется начинать новые строки в первом столбце.

Что должно находиться в файле `main.cf`

Файл `main.cf` может содержать свыше 300 параметров. Тем не менее, организации среднего размера понадобится только несколько параметров, поскольку в большинстве ситуаций для них сгодятся стандартные установки. Автор Postfix настоятельно рекомендует включать в конфигурацию только те параметры, которые имеют нестандартные значения. Таким образом, если в будущем изменится стандартное значение параметра, ваша конфигурация автоматически примет новое значение.

Образец файла `main.cf`, включенный в дистрибутив, содержит множество закомментированных параметров-образцов и краткую документацию. Исходную версию лучше оставить неизмененной в качестве справочной. Для вашей конфигурации лучше всего начать с пустого файла, чтобы ваши настройки не были потеряны в море комментариев.

Базовые настройки

Давайте начнем с наиболее простой конфигурации — с пустого файла. Как ни странно, это самая приемлемая конфигурация Postfix. Она приводит к тому, что создается почтовый сервер, осуществляющий локальную доставку почты в пределах одного и того же домена как локальный хост, и который посылает любые сообщения, направляемые по нелокальным адресам, прямо на соответствующие дистанционные серверы.

Другим примером простой конфигурацией является "null-клиент" (пустой клиент). Это система, которая не осуществляет локальную доставку почты, а направляет исходящую почту на обозначенный центральный сервер. При этой конфигурации определяется несколько параметров, начиная с параметра `mydomain`, который определяет доменную часть имени компьютера, и параметра `myorigin`, который определяет почтовый домен, добавляемый к неполным почтовым адресам. Если параметры `mydomain` и `myorigin` являются одинаковыми, мы можем записать примерно следующее:

```
mydomain = cs.colorado.edu
myorigin = $mydomain
```

Другим параметром, который мы должны задать, является параметр `mydestination`, определяющий локальные почтовые домены. (Они известны также как "канонические" домены.) Если почтовым доменом в адресе получателя сообщения является `mydestination`, сообщение будет доставлено соответствующему пользователю посредством программы `local` (здесь предполагается, что не будет найдено ни одного уместного псевдонима или файла `.forward`). Если в параметре `mydestination` будет указано несколько почтовых доменов, они будут рассматриваться как псевдонимы одного и того же домена.

Нашему null-клиенту не нужна локальная доставка, поэтому этот параметр должен быть пустым:

```
mydestination =
```

Наконец, параметр `relayhost` сообщает Postfix о том, что все нелокальные сообщения нужно посылать на указанный компьютер, а не по их адресам назначения:

```
relayhost = [mail.cs.colorado.edu]
```

Квадратные скобки говорят Postfix о том, что указанную строку нужно обрабатывать как имя компьютера (запись DNS A), а не как имя почтового домена (запись DNS MX).

Так как null-клиенты не должны получать почту из других систем, последнее, что мы должны сделать в конфигурации null-клиента — это закомментировать строку `smtpd` в файле `master.cf`. Благодаря этому Postfix вообще не будет запускать программу `smtpd`. Итак, с помощью всего лишь нескольких строк мы определили полностью функциональный null-клиент!

Для "настоящего" почтового сервера вам понадобятся дополнительные конфигурационные опции, а также некоторые таблицы преобразований. О них мы поговорим в следующих разделах.

Использование `postconf`

`postconf` — это удобное инструментальное средство, которое помогает конфигурировать Postfix. Если его запустить без аргументов, оно напечатает все параметры в их текущей конфигурации. Если вы передадите `postconf` имя какого-нибудь параметра,

она напечатает значение этого параметра. Опция -d заставит postconf напечатать стандартные настройки, а не текущие сконфигурированные значения. Например:

```
$ postconf mydestination
mydestination =

$ postconf -d mydestination
mydestination = $myhostname, localhost.$mydomain, localhost
```

Другой полезной опцией является -n — она заставляет postconf печатать только те параметры, которые отличаются от стандартных. Если вы попросите помочь вам в списке рассылки Postfix, то в письме вы должны будете написать эту конфигурационную информацию.

Таблицы преобразований

Многие аспекты поведения Postfix определяются за счет использования таблиц поиска, которые могут отображать ключи в виде значений или реализовывать простые списки. Например, стандартной настройкой таблицы alias_maps является следующая:

```
alias_maps = dbm:/etc/mail/aliases, nis:mail.aliases
```

Источники данных определяются посредством записи *тип*:*путь*. Обратите внимание на то, что данная конкретная таблица на самом деле использует одновременно два разных источника информации: базу данных dbm и карту NIS. Множество значений можно разделять запятыми, пробелами или путем комбинирования этих символов. В табл. 18.22 перечислены доступные источники данных; эту же информацию можно вывести с помощью команды postconf -m.

Таблица 18.22. Источники информации, используемые в качестве таблиц преобразований Postfix

Тип	Описание
dbm/sdbm	Традиционный файл базы данных dbm или gdbm
cidr	Сетевые адреса в форме CIDR
hash/btree	Файл хэшированной таблицы Berkley DB (замена dbm) или B-дерева
ldap	Служба каталогов LDAP
mysql	База данных MySQL
nis	Служба каталогов NIS
pcre	Регулярные выражения, совместимые с Perl
pgsql	База данных PostgreSQL
proxy	Доступ посредством proxymap (например, чтобы не изменять корневой каталог)
regexp	Регулярные выражения POSIX
static	Возвращает значение, указанное в *путь*, независимо от ключа
unix	Файлы Linux /etc/passwd и /etc/group; использует синтаксис NIS[a]

[a] unix:passwd.byname — это файл passwd, а unix:group.byname — это файл group.

Использовать типы dbm и sdbm нужно только для обеспечения совместимости с традиционной таблицей псевдонимов программы sendmail. Berkley DB (hash) — это более

современная реализация; она более безопасная и более быстрая. Если с совместимостью проблем нет, используйте следующие записи:

```
alias_database = hash:/etc/mail/aliases
alias_maps = hash:/etc/mail/aliases
```

alias_database определяет таблицу, которую заново создает программа newaliases и которая должна соответствовать таблице, которую вы определите в alias_maps. Необходимость в двух параметрах объясняется тем, что alias_maps может включать источники, отличные от DB, такие как mysql или nis, которые не нужно создавать заново.

Все таблицы класса DB (dbm, sdbm, hash и btree) основаны на текстовом файле, который компилируется в бинарный формат, обеспечивающий эффективный поиск. Синтаксис этих текстовых файлов похож на синтаксис конфигурационных файлов применительно к строкам комментариев и строкам-продолжениям. Записи определяются в виде простых пар "ключ/значение", что обеспечивает совместимость с sendmail. Например, следующие строки соответствуют таблице псевдонимов:

```
postmaster:          david, tobias
webmaster:           evi
```

В качестве другого примера мы приведем таблицу доступа для ретрансляции почты, поступающей от любого клиента, имя компьютера которого заканчивается на cs.colorado.edu.

```
.cs.colorado.edu    OK
```

Текстовые файлы компилируются в их бинарные форматы с помощью команды postmap для обычных таблиц и команды postalias для таблиц псевдонимов. Определение таблицы (включая тип) должно стоять в первом аргументе. Например:

```
$ postmap hash:/etc/postfix/access
```

postmap может также запрашивать значения в таблице преобразований:

```
$ postmap -q blabla hash:/etc/postfix/access
$ postmap -q .cs.colorado.edu hash:/etc/postfix/access
OK
```

Локальная доставка

Программа local отвечает за доставку почты в канонический домен. Кроме того, она обрабатывает локальное использование псевдонимов.

Например, если mydestination задать как cs.colorado.edu, и будет поступать почта для получателя evi@cs.colorado.edu, программа local сначала произведет поиск в таблицах alias_maps, а затем рекурсивным образом заменит любые совпадающие записи.

Если подходящий псевдоним не будет найден, local произведет поиск файла .forward в домашнем каталоге пользователя evi и выполнит инструкции, содержащиеся в этом файле (если он существует). (Синтаксис такой же, как и в правой части карты псевдонимов.) Наконец, если файл .forward будет найден, почта будет доставлена в почтовый ящик пользователя evi.

По умолчанию программа local производит запись в стандартные файлы формата mbox в каталоге /var/mail. Это поведение можно изменить посредством параметров, перечисленных в табл. 18.23.

Таблица 18.23. Параметры для локальной доставки в почтовый ящик (установлены в конфигурационном файле `main.cf`)

Параметр	Описание
`mail_spool_directory`	Доставляет почту в центральный каталог, обслуживающий всех пользователей
`home_mailbox`	Доставляет почту в каталог `~`*пользователь* по определенному относительному пути
`mailbox_command`	Доставляет почту с помощью внешней программы, обычно `procmail`
`mailbox_transport`	Доставляет почту посредством службы, определенной в файле `master.cf`[a]
`recipient_delimiter`	Разрешает расширенные имена пользователей (см. описание ниже)

[a] Эта опция связана с серверами почтовых ящиков, такими как Cyrus `imapd`.

Опции `mail_spool_directory` и `home_mailbox` обычно генерируют почтовые ящики формата `mbox`, однако они могут также создавать почтовые ящики `Maildir` в стиле программы `qmail`. Чтобы сделать это, в конце путевого имени нужно добавить косую черту.

Если опции `recipient_delimiter` присвоить значение `+`, почта, адресованная пользователю `evi+`*что-либо*`@cs.colorado.edu`, будет принята для доставки учетной записи `evi`. Благодаря этому пользователи могут создавать специальные адреса и сортировать их почту по адресам назначения. Postfix сначала попытается произвести поиск полного адреса, и только если он окажется неудачным, выделит расширенные компоненты и вернется к базовому адресу. Postfix также произведет поиск подходящего файла переадресации (`.forward+`*что-либо*) для дальнейшего использования псевдонимов.

Виртуальные домены

Если вам нужно обслуживать почтовый домен на вашем почтовом сервере Postfix, вы можете сделать это тремя способами:

- Указать домен в параметре `mydestination`. Доставка выполняется по вышеприведенной схеме: псевдонимы расширяются и почта доставляется соответствующим пользователям.
- Указать домен в параметре `virtual_alias_domains`. В этом случае домен получает свое собственное адресное пространство имен, которое не зависит от системных учетных записей пользователей. Нужно обеспечить возможность преобразования адресов внутри домена (с помощью карт) в реальные адреса вне домена.
- Указать домен в параметре `virtual_mailbox_domains`. Как и в случае с параметром `virtual_alias_domains`, домен имеет свое собственное пространство имен. Однако почту можно доставлять во все почтовые ящики по определенному пути, независимо от учетных записей пользователей.

Домен можно указывать только в одном из этих трех мест. Искомый параметр следует выбирать обдуманно, так как от этого выбора будут зависеть многие элементы конфигурации. Обработку метода `mydestination` мы уже рассмотрели. Теперь перейдем к оставшимся вариантам.

Виртуальные домены псевдонимов

Если домен указан в виде значения параметра `virtual_alias_domains`, то почта, направляемая в данный домен, поступает в программу Postfix и должна быть направлена действительному получателю на локальный компьютер или куда-либо еще.

Пересылка по адресам в виртуальном домене должна быть определена в таблице преобразований, указанной в параметре `virtual_alias_maps`. Записи в таблице имеют адрес в виртуальном домене с левой стороны и действительный пункт назначения с правой стороны.

Неточное имя с правой стороны интерпретируется как имя пользователя на локальном компьютере.

Рассмотрим следующий пример из файла `main.cf`:

```
myorigin = cs.colorado.edu
mydestination = cs.colorado.edu
virtual_alias_domains = admin.com
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

В файле `/etc/mail/admin.com/virtual` могут быть следующие строки:

```
postmaster@admin.com   evi, david@admin.com
abuse@admin.com        evi
david@admin.com        david@schweikert.ch
evi@admin.com          evi
```

Почта для пользователя `evi@admin.com` может быть направлена пользователю `evi@cs.colorado.edu` (с добавлением `myorigin`) и, в конечном счете, окажется в почтовом ящике пользователя evi, поскольку часть `cs.colorado.edu` содержится в параметре `mydestination`.

Определения могут быть рекурсивными: справа могут содержаться адреса, которые в дальнейшем будут определены с левой стороны. Следует иметь в виду, что справа может находиться только список адресов. Если вам нужно выполнить внешнюю программу или использовать файлы `:include`, почту нужно направлять по псевдонимам, которые затем могут быть расширены в соответствии с вашими потребностями.

Чтобы все хранилось в одном файле, можно присвоить параметру `virtual_alias_domain` ту же таблицу преобразований, что и в параметре `virtual_alias`, и поместить специальную запись в таблицу, чтобы отметить ее как виртуальный домен псевдонимов. В файле `main.cf`:

```
virtual_alias_domains = $virtual_alias_maps
virtual_alias_maps = hash:/etc/mail/admin.com/virtual
```

В файле `/etc/mail/admin.com/virtual`:

```
admin.com              notused
postmaster@admin.com   evi, david@admin.com
...
```

Правая сторона записи для почтового домена (admin.com) на самом деле никогда не используется; admin.com присутствует в таблице как независимая запись — этого достаточно, чтобы Postfix считала ее виртуальным доменом псевдонимов.

Виртуальные домены почтовых ящиков

Домены, перечисленные в параметре `virtual_mailbox_domains`, подобны локальным (каноническим) доменам, однако управление списком пользователей и их соответствующими почтовыми ящиками должно осуществляться независимо от системных учетных записей пользователей.

Параметр `virtual_mailbox_maps` указывает на таблицу, которая перечисляет всех действительных пользователей в домене. Карта имеет следующий формат:

пользователь@домен */путь/к/почтовому/ящику*

Если путь заканчивается косой чертой, почтовые ящики будут храниться в формате `Maildir`. Значение параметра `virtual_mailbox_base` всегда стоит перед указанными путями.

Зачастую бывает необходимо назначить псевдонимы для некоторых адресов в параметре `virtual_mailbox_domain`. Для этой цели нужно использовать параметр `virtual_alias_map`. Ниже представлен полный пример.

Файл `main.cf`:

```
virtual_mailbox_domains = admin.com
virtual_mailbox_base = /var/mail/virtual
virtual_mailbox_maps = hash:/etc/mail/admin.com/vmailboxes
virtual_alias_maps = hash:/etc/mail/admin.com/valiases
```

Файл `/etc/mail/admin.com/vmailboxes` может содержать такие записи:

```
evi@admin.com            nemeth/evi/
```

Файл `/etc/mail/admin.com/valiases` может содержать следующие записи:

```
postmaster@admin.com     evi@admin.com
```

Карты виртуальных псевдонимов можно использовать даже в тех адресах, которые не являются доменами виртуальных псевдонимов. Карты виртуальных псевдонимов позволяют осуществлять пересылку на любой адрес из любого домена, независимо от его типа (канонический, виртуальный псевдоним или виртуальный почтовый ящик). Так как пути к почтовым ящикам могут находиться справа от карты виртуальных псевдонимов, использование этого механизма является единственным способом назначения псевдонимов в данном домене.

Управление доступом

Почтовые серверы должны осуществлять ретрансляцию почты для сторонних адресов только по поручению надежных клиентов. Если почтовый сервер направляет почту, поступающую от неизвестных клиентов, на другие серверы, это называется открытой ретрансляцией, не сулящей ничего хорошего. Более подробно об этом можно прочитать в разделе 18.10.

К счастью, Postfix по умолчанию не действует как открытый ретранслятор. В действительности, его стандартные настройки являются довольно ограничивающими. Вполне возможно, что вам придется ослабить ограничения, а не усиливать их. Управление доступом для транзакций типа SMTP в Postfix конфигурируется посредством “списков ограничения доступа” (access restriction list). Параметры, перечисленные в табл. 18.24, определяют, что будет проверяться во время различных фаз сеанса SMTP.

Таблица 18.24. Параметры Postfix для ограничения доступа, производимого посредством SMTP

Параметр	Когда применяется
smtpd_client_restrictions	При запросе на соединение
smtpd_helo_restrictions	Во время выполнения команды HELO/EHLO (начало сеанса)
smtpd_sender_restrictions	Во время выполнения команды MAIL FROM (спецификация отправителя)
smtpd_recipient_restrictions	Во время выполнения команды RCPT TO (спецификация получателя)
smtpd_data_restrictions	Во время выполнения команды DATA (тело сообщения)
smtpd_etrn_restrictions	Во время выполнения команды ETRN[a]

[a] Это специальная команда. Она используется для повторной отправки сообщений, находящихся в очереди.

Самым важным параметром является smtp_recipient_resctrictions, поскольку управление доступом легче всего выполнять, когда известен адрес получателя, и когда он можно установить, является он локальным, или нет. Все остальные параметры в табл. 18.24 в стандартной конфигурации являются пустыми. Стандартным значением является следующее:

```
smtpd_recipient_restrictions = permit_mynetworks, reject_unauth_destination
```

Каждое из определенных ограничений проверяется до тех пор, пока не будет достигнуто окончательное решение относительно того, что нужно делать с почтой. В табл. 18.25 перечислены наиболее часто используемые ограничения.

Таблица 18.25. Обычные ограничения доступа, которые могут быть реализованы в Postfix

Ограничение	Его действие
check_client_access	Проверяет адрес компьютера клиента с помощью таблицы преобразований
check_recipient_access	Проверяет почтовый адрес получателя с помощью таблицы преобразований
permit_mynetworks	Предоставляет доступ к адресам, указанным в параметре mynetworks
reject_unauth_destination	Отклоняет почтовые сообщения, поступающие для нелокальных получателей; ретрансляция не производится

С помощью этих ограничений можно проверить все, а не только определенную информацию наподобие адреса отправителя в параметре smtp_sender_restrictions. Таким образом, для простоты можно поместить все ограничения в один параметр, smtpd_sender_restrictions, поскольку это единственный параметр, который может проверить все (кроме части DATA).

В параметре smtpd_recipient_restriction производится проверка ретрансляции почты. Отключать ограничение reject_unauth_destination не следует, а перед ним нужно выбирать “разрешающие” ограничения.

Таблицы доступа

Каждое ограничение возвращает одно из действий, перечисленных в табл. 18.26. В таких ограничениях, как check_client_access и check_recipient_access, исполь-

зуются таблицы доступа для выбора действия на основании адреса компьютера клиента или адреса получателя, соответственно.

Таблица 18.26. Действия для таблиц доступа

Действие	Назначение
4*nn текст*	Возвращает код временной ошибки 4nn и текстовое сообщение
5*nn текст*	Возвращает код постоянной ошибки 5nn и текстовое сообщение
DEFER_IF_PERMIT	Если ограничения приводят к разрешению (PERMIT), изменяет на временную ошибку
DEFER_IF_REJECT	Если ограничения приводят к отклонению (REJECT), изменяет на временную ошибку
DISCARD	Принимает сообщения и “молча” отклоняет их
DUNNO	Делает вид, что ключ не был найден; проверяет последующие ограничения
FILTER *транспорт*:*назначение*	Передает почту посредством фильтра *транспорт*:*назначение*[a]
HOLD	Блокирует почту в очереди
OK	Принимает почту
PREPEND *заголовок*	Добавляет заголовок в сообщение
REDIRECT *адрес*	Направляет эту почту по указанному адресу
REJECT	Отклоняет почту
WARN *сообщение*	Регистрирует данное предупреждающее сообщение в журналах

[a] См. раздел с описанием обработки спама и вирусов в разделе 18.16.

В качестве примера предположим, что вам нужно разрешить ретрансляцию для всех компьютеров в домене cs.colorado.edu, и что вам нужно разрешить отправку писем во внутренний список рассылки newsletter@cs.colorado.edu только надежным пользователям. Эти политики можно реализовать с помощью следующих строк в конфигурационном файле main.cf:

```
smtpd_recipient_restrictions =
       check_client_access hash:/etc/postfix/client_access
       check_recipient_access hash:/etc/postfix/recipient_access
       reject_unauth_destination
```

Обратите внимание на то, что при определении списка значений для параметра запятые ставить не обязательно.

В файле /etc/postfix/client:

```
.cs.colorado.edu     OK
```

В файле /etc/postfix/recipient_access:

```
newsletter@cs.colorado.eduREJECT Internal list
```

Текст после REJECT является необязательной строкой — он отправляется клиенту вместе с кодом ошибки. Он говорит отправителю, по какой причине было отклонено письмо.

Аутентификация клиентов

Для тех пользователей, которые отправляют почту из своего дома, самым простым является способ маршрутизации через почтовый сервер поставщика услуг Internet, независимо от того, какой адрес отображается в электронном письме. Большинство поставщиков услуг Internet доверяют своим прямым клиентам и разрешают ретрансляцию. Если эта конфигурация невозможна, или если вы применяете такую систему, как Sender ID или SPF, убедитесь в том, что мобильные пользователи смогут пройти процедуру проверки подлинности за пределами вашей сети для подачи сообщений вашей программе `smtpd`.

Эту проблему можно решить следующим способом: нужно скомбинировать протоколы проверки подлинности, используемые протоколами POP или IMAP. Пользователи, которым необходимо отправить почту, должны также иметь возможность читать ее, поэтому одна процедура проверки подлинности может служить для каждого из этих процессов.

Комбинированная система, называемая "POP перед SMTP" или "IMAP перед SMTP", работает примерно так: как только пользователь прошел проверку подлинности, выполненную демоном POP или IMAP, IP-адрес пользователя явным образом предъявляется демону SMTP приблизительно на 30 минут. База данных таких клиентов поддерживается демоном, специально предназначенным для выполнения этой задачи. Конфигурация в Postfix выглядит примерно следующим образом:

```
smtpd_recipient_restrictions = permit_mynetworks
        check_client_access hash:/etc/postfix/pop-before-smtp
        reject_unauth_destination
```

Для решения этой проблемы лучше всего использовать механизм SMTP AUTH для проверки подлинности прямо на уровне SMTP. Чтобы это решение заработало, Postfix необходимо скомпилировать с поддержкой библиотеки SASL. После этого можно будет использовать следующую конфигурацию:

```
smtpd_sasl_auth_enable = yes
smtpd_recipient_restrictions = reject_non_fqdn_recipient
        permit_mynetworks
        permit_sasl_authenticated
```

Вам придется также поддерживать зашифрованные соединения, чтобы избежать отправки пароля в открытом виде. В версиях Postfix, вышедших после версии 2.2, нужно установить заплату исходного кода с помощью заплаты Postfix/TLS. В конфигурационный файл `main.cf` нужно добавить следующие строки:

```
smtpd_use_tls = yes
smtpd_tls_auth_only = yes
smtpd_tls_cert_file = /etc/certs/smtp.pem
smtpd_tls_key_file = $smtpd_tls_cert_file
```

Борьба со спамом и вирусами

Postfix имеет много функциональных средств, которые могут помочь блокировать подозрительные почтовые сообщения.

Одной из мер защиты является строгая реализация протокола SMTP. Легитимные почтовые серверы должны принимать этот протокол, а отправители спама и вирусов часто не принимают его, тем самым обнаруживая себя. К сожалению, сейчас существуют злоумышленные почтальоны, которые могут обрабатывать легитимную почту, по-

этому эта технология не является достаточно надежной. Нужно тщательно подбирать ограничения и следить за журнальными файлами. В табл. 18.27 показаны некоторые из функциональных особенностей, относящихся к этой категории.

Таблица 18.27. Параметры и ограничения для строгой проверки с использованием протокола SMTP

Опция	Назначение

`reject_non_fqdn_sender`
`reject_non_fqdn_recipient`
`reject_non_fqdn_hostname`
	Отклоняет сообщения, содержащие неточный домен отправителя, домен получателя или имя компьютера HELO/EHLO (ограничение)

`reject_unauth_pipelining`	Прерывает текущий сеанс, если клиент не ожидает получения информации о состоянии команды, прежде чем продолжить сеанс (ограничение)
`reject_unknown_sender_domain`	Отклоняет сообщения, имеющие нераспознаваемый домен отправителя (ограничение)[a]
`smtpd_helo_required`	Запрашивает HELO/EHLO в начале переговоров (параметр, или yes, или no)
`strict_rfc821_envelopes`	Требует правильного синтаксиса для адресов электронной почты в командах MAIL FROM и RCPT TO (параметр, или yes, или no)

[a] Возвращает временное сообщение об ошибке, поскольку проблема может быть вызвана кратковременным сбоем в системе DNS.

Чтобы проверить ограничение перед его вводом в действие (между прочим, это нужно делать всегда), вставьте перед ним ограничение `warn_if_reject`, чтобы преобразовать полное отклонение в предупреждающее журнальное сообщение.

Черные списки

Как и в случае с программой `sendmail`, вы можете научить Postfix проверять входящую почту с помощью черных списков, основанных на информации о DNS; более подробно о черных списках можно прочитать в разделе “Черные списки” ранее в этой главе. Чтобы разрешить это поведение, используйте ограничение `reject_rbl_client`, после которого нужно указать адрес DNS-сервера. Подобным образом ведет себя и ограничение `reject_rhsbl_sender` — оно проверяет доменное имя адреса отправителя, а не имя компьютера клиента.

В следующем примере показана относительно полная конфигурация для борьбы со спамом в файле `main.cf`:

```
strict_rfc821_envelopes = yes
smtpd_helo_required = yes
smtpd_recipient_restrictions =reject_unknown_sender_domain
         reject_non_fqdn_sender
         reject_non_fqdn_recipient
         permit_mynetworks
         check_client_access hash:/etc/postfix/client_access
         reject_unauth_destination
         reject_unauth_pipelining
         reject_rbl_client relays.ordb.org
         reject_rhsbl_sender dsn.rfc-ignorant.org
```

Обратите внимание на то, что мы поместили некоторые ограничения перед `permit_mynetworks`. Это будет гарантировать то, что наши клиенты отправляют правильно отформатированную почту. Благодаря этому можно легко обнаружить ошибки в конфигурации.

SpamAssassin и `procmail`

Postfix поддерживает SpamAssassin и другие фильтры подобного рода. SpamAssassin рассматривается в конце раздела 18.10, а общая информация об этих инструментальных средствах содержится в разделе "Средство `local_procmail`" ранее в этой главе.

`procmail` можно запускать из пользовательских файлов `.forward`, однако этот путь чересчур сложный и сопряжен с ошибками. Более удачный подход заключается в том, чтобы поместить в файл `main.cf` следующую строку:

```
mailbox_command = /usr/bin/procmail -a "$EXTENSION"
```

После этого Postfix будет использовать программу `procmail` для доставки почты, а не записывать сообщения прямо в почтовый спул. Аргументы, заданные в `procmail`, передают адресное расширение (часть после знака +); доступ к нему в `procmail` можно получить через сочетание `$1`.

Демоны политики

В версии Postfix 2.1 предложен механизм делегирования управления доступом внешним программам. Эти программы, называемые демонами политики, получают всю информацию, которой Postfix располагает в отношении почтового сообщения, и должны вернуть одно из решений, перечисленных в табл. 18.26.

Пожалуй, самой интересной особенностью, которую можно реализовать благодаря демону политики, является создание "серых" списков (greylisting). При создании серых списков классифицируется каждое входящее сообщение на основе имени компьютера клиента, адресе отправителя и адресе получателя. В первый раз, при получении этих трех показателей Postfix возвращает отправителю код временной ошибки. Легитимные почтовые серверы пытаются повторно осуществить доставку в течение 10 минут после этого — это то время, когда сообщение может быть принято. Поскольку первая попытка повторной доставки сообщения обычно производится в течение одной минуты, письмо долго не задерживается.

По своей сути эта процедура похожа на работу тех функциональных особенностей Postfix, которые требуют строгого соответствия с протоколом SMTP. В случае создания серых списков попытка повторной отправки письма сама по себе свидетельствует о том, что мы имеем дело с легитимным почтовым сервером. С помощью серых списков можно эффективно бороться со спамом, так как многие источники спама пользуются программным обеспечением, которое не осуществляет повторную отправку сообщений.

Фильтрация содержимого

Для проверки заголовков и тел электронных сообщений на предмет "контрабанды" Postfix может использовать регулярные выражения. Postfix может также передавать сообщения другим программам, таким как специальные средства для борьбы со спамом или антивирусные приложения.

Проверки заголовков и тел сообщений выполняются в реальном времени по мере того, как сообщения принимаются через SMTP. Каждое проверяемое регулярное выражение генерирует действие, перечисленное в таблице 18.26, в случае совпадения регулярного выражения. Например, строка

```
header_checks = regexp:/etc/postfix/header_checks
```

в файле `main.cf`, а также следующая строка в файле `/etc/postfix/header_checks`

```
/^Subject: reject-me/ REJECT You asked for it
```

отклонит любое сообщение, тема которого начинается со слов “reject-me”. Несмотря на то что поддержка регулярного выражения всегда обеспечивается на высоком уровне, при обработке электронных сообщений может возникать множество трудностей. В частности, этот метод не является эффективным для борьбы со спамом или вирусами.

Эффективная фильтрация вирусов обычно реализуется посредством Perl-программы Amavis, которая связывает программное обеспечение почтового сервера с одним или несколькими антивирусными приложениями. Такие фильтры конфигурируются с помощью параметра `content_filter`, который сообщает Postfix о том, что каждое входящее сообщение нужно пропускать через указанную службу. Помимо задания параметра `content_filter` вам нужно будет изменить некоторые существующие записи в файле `master.cf` и добавить некоторые новые записи. Amavis содержит подробные инструкции относительно выполнения этой процедуры. На данный момент разработано много вариантов программы Amavis; мы рекомендуем `amavisd-new`, написанную Марком Мартинеком (Mark Martinec).

Отладка

Если у вас возникнет проблема с работой Postfix, проверьте сначала ее журнальные файлы. Наверняка все ответы на ваши вопросы можно будет найти прямо в них, нужно лишь уметь это делать. Каждая программа Postfix обычно регистрирует запись в журнале для каждого обрабатываемого сообщения. Например, ниже показаны строки для исходящего сообщения:

```
Aug 18 22:41:33 nova postfix/pickup: 0E4A93688: uid=506
   from=<dws@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/cleanup: 0E4A93688: message-id=
   <20040818204132.GA11444@ee.ethz.ch>
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: from=<dws@ee.ethz.ch>,
   size=577,nrcpt=1 (queue active)
Aug 18 22:41:33 nova postfix/smtp: 0E4A93688:
   to=<evi@ee.ethz.ch>,relay=tardis.ee.ethz.ch[129.132.2.217],delay=0,
   status=sent (250 Ok: queued as 154D4D930B)
Aug 18 22:41:33 nova postfix/qmgr: 0E4A93688: removed
```

Как можно видеть, во многих строках присутствуют интересные сведения. Обратите внимание на то, что идентификатор `0E4A93688` является общим для каждой строки: Postfix присваивает идентификатор очереди по мере того, как сообщение попадает в почтовую систему, и никогда не изменяет его. Таким образом, при поиске в журнале истории сообщений в первую очередь нужно сосредоточиться на определении идентификатора очереди, присвоенного сообщению. Когда вы будете знать его, вам будет легче выполнять команду `grep` в журналах для работы с релевантными записями.

Postfix хорошо справляется с задачей регистрации полезных сообщений о выявляемых проблемах. Однако иногда бывает трудно найти важные строки среди тысяч обычных сообщений о состоянии. Здесь пригодятся некоторые инструментальные средства, описанные в разделе 10.5.

Просмотр очереди

Другим местом, в котором можно выявить проблемы, является очередь сообщений. Как и в системе `sendmail`, команда `mailq` печатает содержимое очереди. Ее можно использовать для того, чтобы выяснить причину возникновения проблем с данным сообщением.

Еще одним полезным инструментальным средством является сценарий `qshape`, который входит в состав последних версий Postfix. Он показывает сводную статистику о содержимом очереди. Результаты выполнения этого сценария могут выглядеть примерно так:

```
# qshape deferred
                  T  5 10 20 40 80 160 320  640 1280 1280+
          TOTAL  78  0  0  0  7  3   3   2   12    2    49
       expn.com  34  0  0  0  0  0   0   0    9    0    25
   chinabank.ph   5  0  0  0  1  1   1   2    0    0     0
prob-helper.biz   3  0  0  0  0  0   0   0    0    0     3
```

Сценарий `qshape` подводит итог данной очереди (в данном случае это отложенная очередь), в которой выполнена сортировка по доменам получателей. В столбцах содержится отчет о количестве минут, в течение которых связанные сообщения находятся в очереди. Например, можно видеть, что 49 сообщений, адресованных домену expn.com, находятся в очереди более 1280 минут. Все адреса назначения в данном примере наводят на мысль о том, что сообщения отправлялись сценариями в ответ на спам.

`qshape` может также подвести итог по домену отправителей — для этого нужно использовать флаг `-s`.

Мягкий рикошет

Если параметру `soft_bounce` присвоить значение `yes`, Postfix будет отправлять временные сообщения об ошибках при каждой отправке постоянных сообщений об ошибках, таких как “user unknown” (пользователь неизвестен) или “relaying denied” (ретрансляция запрещена). Это очень эффективное средство проверки, так как оно позволяет отслеживать расположение сообщений после изменения в конфигурации без риска навсегда потерять легитимные сообщения. Все, что вы отклоните, в конечном счете вернется отправителю для повторной отправки. Не забывайте отключать эту возможность после завершения проверки, иначе вы столкнетесь с тем, что каждое сообщение будет отклоняться снова и снова.

Проверка управления доступом

Чтобы проверить ограничения в управлении доступом, проще всего отправить сообщение не из данного компьютера, а затем посмотреть, что при этом произойдет. Эту проверку хорошо выполнять в качестве базовой, однако она не охватывает особые условия (например, если письма были отправлены из определенного домена, в котором вы не регистрировались).

В версии Postfix 2.1 предложено расширение протокола SMTP, называемое XCLIENT, которое имитирует подачу сообщений из другого места. По умолчанию эта особенность не используется, однако если в конфигурационном файле `main.cf` включить следующую строку, вы сможете включить эту особенность для тех сообщений, которые будет отправлять `localhost`:

```
smtpd_authorized_xclient_hosts = localhost
```

Сеанс проверки может выглядеть примерно так:

```
$ telnet localhost 25
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 tardis.ee.ethz.ch ESMTP Postfix
XCLIENT NAME=mail.cs.colorado.edu ADDR=192.168.1.1
250 Ok
HELO mail.cs.colorado.edu
250 tardis.ee.ethz.ch
MAIL FROM: <evi@colorado.edu>
250 Ok
RCPT TO: <david@colorado.edu>
554 <david@colorado.edu>: Relay access denied
```

18.17. Рекомендуемая литература

- Costales, Bryan, and Eric Allman. *sendmail (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2002.

  Эта книга представляет собой основательный труд — она содержит 1200 страниц. В ней вы найдете учебное пособие и полный справочный раздел. Изучать эту книгу можно, открывая ее на случайной странице — на наш взгляд, это можно отнести к достоинствам справочника. Также в ней есть хороший предметный указатель.

- Clayton, Richard. *Good Practice for Combating Unsolicited Bulk Email*. RIPE/Demon Internet. 2000, `www.ripe.net/ripe/docs/ripe-206.html`

  Этот документ предназначен для Internet-провайдеров. В нем содержится множество информации о политике, а также полезные ссылки на технические темы.

- Schwartz, Alan. *SpamAssassin*. Sebastopol, CA: O'Reilly Media, 2005.
- Schwartz, Alan, and Paula Ferguson. *Managing Mailing Lists*. Sebastopol, CA: O'Reilly Media, 1998.
- Hazel, Philip. *The Exim Smtp Mail Server: Official Guide for Release 4*. Cambridge, UK: User Interface Technologies, Ltd., 2003.

Документацию по Exim и информацию можно на сайте `www.exim.org`.

На `man`-странице программы `sendmail` описаны ее аргументы командной строки. Обзор можно найти в книге *Sendmail: An Internetwork Mail Router*, автор Эрик Оллмен (Eric Allman).

Инструкции по инсталляции и хорошее описание конфигурационного файла рассмотрены в руководстве *Sendmail Installation and Operation Guide*, которое можно найти в подкаталоге `doc/op` дистрибутива `sendmail`. Этот документ является относительно полным, а вместе с файлом `README` из каталога `cf` он дает хороший обзор системы `sendmail`.

На сайтах `www.sendmail.org`, `www.sendmail.org/~ca` и `www.sendmail.org/~gshapiro` можно найти документы, касающиеся `sendmail`, ответы на вопросы и учебные пособия.

В документе RFC2822, который заменяет документ RFC822, дан синтаксис сообщений и адресов в сетевых почтовых системах, а в документе RFC1123 описаны требования к компьютерам. По сути, это функциональные спецификации, на основе которых построена `sendmail`.

Документ RFC2821, заменяющий документ RFC821, определяет протокол SMTP, а документы RFC1869, RFC1870, RFC1891 и RFC1895 расширяют его до ESMTP.

Документ RFC974 описывает записи MX в системе доменных имен (DNS) и их отношение к маршрутизации почты. Ниже перечислены остальные документы RFC, относящиеся к электронной почте.

- RFC1731 — IMAP4 Authentication Mechanisms (Механизмы аутентификации IMAP4).
- RFC1733 — Distributed Electronic Mail Models in IMAP4 (Распределенные модели электронной почты в IMAP4).
- RFC2033 — Local Mail Transfer Protocol (Протокол локальной пересылки почты)
- RFC2076 — Common Internet Message Headers (Общие заголовки сообщений Internet)
- RFC2142 — Mailbox Names for Common Services, Roles and Functions (Имена почтовых ящиков для общих служб, ролей и функций).
- RFC2505 — Anti-Spam Recommendations for SMTP MTAs (Рекомендации по борьбе со спамом для агентов пересылки сообщений SMTP).
- RFC2635 — Guidelines for Mass Unsolicited Mailings (Руководства по массовой рассылке писем по базе адресов электронной почты, добровольно предоставленных пользователями)[35].
- RFC2821 — Simple Mail Transfer Protocol (Протокол SMTP).
- RFC2822 — Internet Message Format (Формат сообщений Internet).
- RFC4405 — SMTP Service Extension for Indicating Message Submitters (Расширение службы SMTP для отображения отправителей сообщений)[35].
- RFC4406 — Sender ID: Authenticating E-Mail (Идентификационный номер отправителя: проверка подлинности электронной почты).
- RFC4408 — SPF for Authorizing Use of Domains in E-Mail, Version 1 (Предпочтительный вариант для легитимного использования доменов в электронной почте, версия 1).
- RFC4409 — Message Submission for Mail (Подача сообщения для отправки по электронной почте).

Документы RFC2821 (SMTP) и RFC2822 (Internet Message Format) заменяют некоторые из наиболее часто просматриваемых документов, касающихся электронной почты; они заменяют документы RFC821, RFC822, RFC974 и RFC1869. Документы RFC2821 и RFC2822 впервые были опубликованы в апреле 2001 г. и являются предложенными стандартами.

[35] Заголовок пересказан автором книги.

18.18. Упражнения

18.1. [специфика sendmail]. Перечислите вкратце различия и схожести между genericstable и virtusertable. В каких ситуациях можно встретить каждый из них?

18.2. [специфика sendmail]. Сравните использование /etc/mail/aliases с использованием сервера LDAP для хранения почтовых псевдонимов. Какие преимущества и недостатки характерны для каждого из этих вариантов?

18.3. Расскажите вкратце о различиях между почтовым пользовательским агентом (MUA), агентом доставки (DA) и агентом доступа (AA). В чем состоят различия между агентом пересылки сообщений (MTA) и агентом подачи почты (MSA).

18.4. [специфика sendmail]. Что собой представляет интерпретатор smrsh? В каких случаях нужно его использовать вместо интерпретатора /bin/sh? Если smrsh используется в вашей организации, то каким программам разрешено работать в качестве почтальона? Имеет ли какой-нибудь из них серьезные проблемы в безопасности?

18.5. [специфика sendmail]. Напишите небольшой файл /etc/mail/aliases, который покажет три разных типа псевдонимов. Расскажите вкратце о том, что делает каждая строка в сценарии и почему она может быть полезной.

☆ 18.6. Опишите вкратце следующий заголовок электронного письма. По какому пути было доставлено это письмо? Кому оно было адресовано, и кому оно было доставлено? Сколько времени письмо шло от отправителя до получателя?

```
From clements@boulderlabs.com Fri Dec 28 17:06:57 2001
Return-Path: <clements@mail.boulderlabs.com>
Received: from boulder.Colorado.EDU (boulder.Colorado.EDU
  [128.138.240.1]) by ucsub.colorado.edu (8.11.6/8.11.2/ITS-5.0/student)
  with ESMTP idfBT06vF10618 for <hallcp@ucsub.Colorado.EDU>; Fri,
  28 Dec 2001 17:06:57-0700 (MST)

Received: from mail.boulderlabs.com (mail.boulderlabs.com
  [206.168.112.48]) by boulder.Colorado.EDU
  (8.10.1/8.10.1/UnixOps+Hesiod (Boulder)) with ESMTP id
  fBT06uL13184; Fri, 28 Dec 2001 17:06:56 -0700 (MST)

Received: from ath.boulderlabs.com (cpe-24-221-212-162.co.sprintbbd.net
  [24.221.212.162]) by mail.boulderlabs.com (8.11.6/8.11.6) with ESMTP
  id fBT06oQ29214 for <booklist@boulderlabs.com>; Fri, 28 Dec 2001
  17:06:50 -0700 (MST) (envelope-from
  clements@mail.boulderlabs.com)

From: David Clements <clements@boulderlabs.com>
Received: (from clements@localhost) by ath.boulderlabs.com
  (8.11.6/8.11.4) id fBT06ma01470 for booklist@boulderlabs.com; Fri,
  28 Dec 2001 17:06:48 -0700 (MST) (envelope-from clements)
Date: Fri, 28 Dec 2001 17:06:48 -0700 (MST)
Message-Id: <200112290006.fBT06ma01470@ath.boulderlabs.com>
To: boolist@boulderlabs.com
Subject: Book Questions
```

☆ 18.7. [специфика `sendmail`]. Перечислите приставки к файлам в каталоге очереди сообщений и объясните, что обозначает каждая из них. Почему важно удалять одни файлы очереди, и ни в коем случае нельзя удалять другие? Как могут использоваться некоторые приставки для устранения ошибок в конфигурации `sendmail`?

18.8. Посмотрите на работу `mailq` в почтовом сервере вашей организации. Есть ли в каталоге какой-нибудь "мусор"? Есть ли в нем какие-нибудь сообщения без управляющих файлов или управляющие файлы без сообщений? Найдите самое "древнее" сообщение в очереди. (Необходим доступ с правами суперпользователя.)

☆ 18.9. [специфика `sendmail`]. Объясните назначение каждого из следующих макросов `m4`. Если макрос включает файл, дайте краткое описание того, каким должно быть содержимое этого файла.

а) `VERSIONID`

б) `OSTYPE`

в) `DOMAIN`

г) `MAILER`

д) `FEATURE`

☆ 18.10. Объясните, что собой представляют записи MX? Почему записи MX являются важными для доставки почты? Дайте пример, в котором неправильно сконфигурированная запись MX может привести к невозможности доставки электронного письма.

☆ 18.11. К каким последствиям может привести попадание адреса в черный список `sbl-xbl.spamhaus.org` или ему подобный? Опишите вкратце некоторые приемы, позволяющие избежать попадания в эти списки.

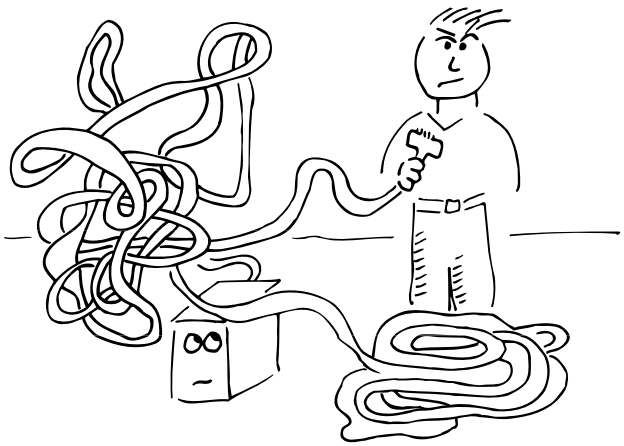
☆ 18.12. Если в вашей организации разрешена работа `procmail`, и если у вас есть разрешения от вашей локальной группы системных администраторов, настройте свой личный конфигурационный файл `procmail` так, чтобы показать, как `procmail` может скомпрометировать защиту.

☆☆ 18.13. Изучите текущую конфигурацию агентов пересылки сообщений (MTA) на вашем сайте. Какие особенности MTA используются? Можете ли вы найти какие-либо проблемы в конфигурации? Каким образом можно улучшить конфигурацию?

☆☆ 18.14. Найдите в своем почтовом ящике сообщение, являющееся спамом (или дождитесь его поступления) и изучите его заголовок. Постарайтесь найти свидетельства того, что данное письмо является спамом. Запустите некоторые из инструментальных средств, рассмотренных в этой главе, (например, SpamCop или SpamAssassin), и составьте отчет об их работе. Каким образом производится распознание сфальсифицированных заголовков? Дайте свое заключение относительно отправителя спама, достоверности перечисленных компьютеров и попытайтесь выявить другие сведения, характеризующие спам.

Глава 19

Управление сетями



Поскольку при объединении компьютеров в сети число связей между ними увеличивается, в сетях, как правило, усугубляются все имеющиеся проблемы. Как гласит народная мудрость, “работа в сети — это когда вы не можете выполнить свое задание из-за отказа компьютера, о котором никогда не слышали”.

Управление сетями — это прежде всего искусство и наука поддержания их в работоспособном состоянии. Сюда обычно входят следующие задачи:

- поиск неисправностей в сетях, шлюзах и важных серверах;
- разработка схем уведомления администратора о наличии проблем;
- общий мониторинг сети с целью распределения нагрузки в ней и планирования ее дальнейшего расширения;
- документирование и визуализация работы сети;
- управление сетевыми устройствами с центральной станции.

В отдельном сегменте сети вряд ли стоит внедрять формальные процедуры управления сетью. Достаточно провести тщательное тестирование сети после ее прокладки и время от времени проверять уровень нагрузки. Сломается — почините.

По мере роста сети процедуры управления должны становиться все более автоматизированными. В сети, где несколько подсетей объединяется посредством коммутаторов или маршрутизаторов, решение административных задач можно автоматизировать с помощью сценариев и простейших программ. Если используются протоколы глобальных сетей или сложные топологии ЛВС, рассмотрите вопрос приобретения выделенных станций управления сетью со специальным программным обеспечением.

В некоторых случаях усложнение системы управления сетью диктуется необходимостью обеспечения надежности. Часто бывает так, что возникновение проблемы в сети приводит к полной остановке работы. Если подобные задержки недопустимы, лучше приобрести и установить высококлассную корпоративную систему управления сетью.

К сожалению, даже самая лучшая система не поможет предупредить все проблемы. Важно иметь хорошо документированную схему сети и высококвалифицированный обслуживающий персонал, способный справляться с неизбежными сбоями.

19.1. Поиск неисправностей в сетях

Существует несколько хороших утилит, позволяющих искать неисправности в сети на уровне TCP/IP. Большинство из них выдает низкоуровневую информацию, поэтому для того чтобы пользоваться ими, нужно хорошо понимать принципы работы протоколов TCP/IP и маршрутизации.

С другой стороны, источником проблем в сети могут служить и ошибки в работе таких высокоуровневых протоколов, как DNS, NFS или HTTP. Прежде чем приступать к чтению этой главы, прочтите главы 12 и 13.

В этом разделе мы рассмотрим общую стратегию поиска неисправностей, после чего перейдем к знакомству с основными инструментами этого поиска: утилитами `ping`, `traceroute`, `netstat`, `tcpdump` и Wireshark. Команда `arp`, которая также предназначена для поиска неисправностей в сети, здесь не описывается; о ней рассказывалось разделе 12.6.

Прежде чем набрасываться на собственную сеть, примите во внимание следующие принципы.

- Вносите пошаговые изменения в конфигурацию сети, тщательно проверяя результаты, чтобы убедиться в совпадении полученного эффекта с ожидаемым. Изменения, которые не дали нужного результата, должны отменяться.
- Задокументируйте возникшую ситуацию и все вносимые в нее изменения.
- Начните с “края” системы или сети и продвигайтесь по ключевым ее компонентам, пока не доберетесь до источника неисправности. Например, начните с исследования сетевой конфигурации клиентского компьютера, затем проверьте физическое соединение клиента с сетью, сетевое оборудование и, наконец, сетевые аппаратные средства сервера и его программную конфигурацию.
- Будьте в курсе событий. Сетевые проблемы оказывают влияние на многих людей: пользователей, провайдеров, системных администраторов, инженеров по телекоммуникациям, сетевых администраторов и т.д. Постоянный контакт с другими специалистами позволит вам не мешать друг другу при решении проблемы.
- Работайте в команде. Многолетний опыт показывает, что люди совершают меньше глупых ошибок, если им оказывают поддержку.
- Помните о многоуровневой структуре сетевых средств. Начните с верхнего или нижнего уровня и последовательно продвигайтесь по стеку протоколов, проверяя их работу.

Последнее замечание заслуживает более подробного рассмотрения. Как указывалось в разделе 12.2, в семействе протоколов TCP/IP определяются уровни абстракции, на которых функционируют различные компоненты сети. Например, протокол HTTP зависит от протокола TCP, который, в свою очередь, основан на протоколе IP, а последний взаи-

модействует с протоколом Ethernet, работоспособность которого зависит от целостности сетевого кабеля. Можно существенно уменьшить время поиска неисправности, если предварительно понять, средства какого уровня ведут себя неправильно.

Проходя стек протоколов уровень за уровнем (вверх или вниз), проверяйте следующее:

- Есть ли физическое соединение и светится ли индикатор связи?
- Правильно ли сконфигурирован интерфейс?
- Отображаются ли в таблицах маршрутизации адреса других компьютеров?
- Находит ли команда `ping` хост `localhost` (127.0.0.1)?
- Находит ли команда `ping` другие компьютеры локальной сети по IP-адресам?
- Правильно ли работает служба DNS[1]?
- Находит ли команда `ping` другие компьютеры локальной сети по именам?
- Находит ли команда `ping` компьютеры в другой сети?
- Работают ли высокоуровневые сетевые службы, такие как Web-и SSH-серверы?

Определив, на каком уровне возникает проблема, не спешите с выводами. Подумайте сначала, как отразятся последующие проверки и вносимые изменения на других сетевых службах и компьютерах.

19.2. Команда `ping`: проверка доступности компьютера

Команда `ping` удивительно проста, но во многих случаях ее оказывается вполне достаточно. Она посылает ICMP-пакет ECHO_REQUEST конкретному компьютеру и ждет ответа. Несмотря на простоту, команда `ping` является одним из основных инструментов отладки сетей.

Команду `ping` можно применять для проверки работоспособности отдельных компьютеров и сегментов сети. В обработке ее запроса участвуют таблицы маршрутизации, физические компоненты сетей и сетевые шлюзы, поэтому для достижения успешного результата сеть должна быть в более или менее рабочем состоянии. Если команда не работает, можно быть совершенно уверенным в том, что более сложные средства тем более не функционируют. Однако это правило неприменимо в сетях, где брандмауэры блокируют эхо-запросы ICMP. Убедитесь в том, что брандмауэр не препятствует работе команды `ping`, прежде чем грешить на зондируемый компьютер, который якобы игнорирует эту команду. В конце концов, отключите на короткое время брандмауэр для проверки работоспособности сети.

Если не задан аргумент “число пакетов”, команда `ping` работает в бесконечном цикле. Чтобы прервать работу команды, нужно нажать специальную клавиатурную комбинацию (каковой обычно является <Ctrl+C>).

Приведем пример:

```
$ ping beast
PING beast (10.1.1.46): 56 bytes of data.
64 bytes from beast (10.1.1.46): icmp_seq=0 ttl=54 time=48.3ms
```

[1] Если компьютер загружается очень медленно, зависает при загрузке или при попытке установить связь посредством утилиты `telnet`, вероятнее всего, служба DNS функционирует неверно.

```
64 bytes from beast (10.1.1.46): icmp_seq=1 ttl=54 time=46.4ms
64 bytes from beast (10.1.1.46): icmp_seq=2 ttl=54 time=88.7ms
^C
--- beast ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2026ms
rtt min/avg/max/mdev = 46.490/61.202/88.731/19.481 ms
```

Информация о компьютере `beast` включает его IP-адрес, порядковый номер ответного ICMP-пакета и полное время прохождения пакета. Полученные результаты свидетельствует о том, что компьютер `beast` работает и подключен к сети.

В нормально функционирующей сети команда `ping` позволяет выяснить, включен компьютер или нет. Если же точно известно, что зондируемый компьютер включен и работает правильно, то с помощью этой команды можно получить полезную информацию о состоянии сети. Пакеты, отправляемые командой `ping`, как и любые другие пакеты, маршрутизируются средствами протокола IP, и успешное получение ответного пакета свидетельствует о том, что все сети и шлюзы между источником и приемником пакета функционируют правильно, хотя бы в первом приближении.

Порядковый номер ICMP-пакета — особенно ценный элемент информации. Отсутствие того или иного номера в последовательности свидетельствует об удалении пакета и обычно сопровождается соответствующим сообщением. Несмотря на то что протокол IP не гарантирует доставку пакетов, пакеты будут пропадать только в том случае, если сеть чересчур загружена. Потерю пакетов следует обязательно выявлять, потому что этот факт иногда скрывается протоколами более высокого уровня. Может показаться, что сеть функционирует нормально, хотя на самом деле она работает гораздо медленнее, чем должна, причем не только из-за повторной передачи пакетов, но также из-за "накладных расходов", требуемых для выявления и обработки пропавших пакетов соответствующими протоколами.

В случае пропадания пакетов сначала выполните команду `traceroute` (описана ниже) для выяснения маршрута, по которому пакеты следуют к компьютеру-адресату. Затем посредством команды `ping` проверьте все промежуточные шлюзы, через которые пролегает маршрут, и узнайте, на каком этапе теряются пакеты. Чтобы точно диагностировать проблему, следует отправить такое количество пакетов, которое позволит получить статистически достоверные результаты. Неисправность сети находится на участке между последним шлюзом, для которого количество потерянных пакетов не больше некоторой заданной величины, и шлюзом, при обращении к которому количество потерянных пакетов превышает эту величину.

Полное время прохождения пакетов, сообщаемое командой `ping`, дает представление об общей скорости передачи пакета по сети. Колебания этого значения, как правило, не являются признаком проблем. Иногда пакеты задерживаются на десятки и сотни миллисекунд без видимой причины — просто так работает протокол IP. Но все-таки следует ожидать, что полное время прохождения пакетов будет более-менее постоянным, за некоторыми исключениями. Большинство современных маршрутизаторов ограничивает скорость выдачи ответов на ICMP-запросы, т.е. маршрутизатор может задержать ответ на пакет команды `ping` в связи с общим ограничением трафика протокола ICMP.

Команда `ping` позволяет отправлять эхо-пакеты любого размера, так что использовав пакет, размер которого превышает значение MTU для данной сети (в частности, 1500 байт в сети Ethernet), можно включить режим принудительной фрагментации. С помощью такого приема можно выявлять ошибки передачи данных в самом кабеле и другие низкоуровневые ошибки типа проблем с перегрузкой сети или проблем с вир-

туальной частной сетью (VPN). Указать желаемый размер пакета в байтах можно при помощи флага -s:

```
$ ping -s 1500 cuinfo.cornell.edu
```

При работе с командой ping учитывайте следующие ограничения.

Во-первых, сложно отличить неисправность сети от неисправности сервера, пользуясь только этой командой. В среде, где проводимые при помощи ping проверки обычно проходят нормально, неудачное выполнение команды ping означает лишь то, что *что-то* не так. (Сетевые брандмауэры иногда намеренно блокируют ICMP-пакеты.)

Во-вторых, успешное выполнение команды ping не гарантирует получение подробной информации о состоянии целевого компьютера. Эхо-запросы обрабатываются в стеке протокола IP и не требуют, чтобы на зондируемом компьютере выполнялся серверный процесс. Получение ответа гарантирует только то, что компьютер включен и ядро системы функционирует. Для проверки работы отдельных служб, таких как HTTP и DNS, следует применять высокоуровневые средства.

19.3. Команда traceroute: трассировка IP-пакетов

Команда traceroute, первоначально написанная Ваном Якобсоном (Van Jacobson), раскрывает последовательность шлюзов, через которые проходит IP-пакет, чтобы достичь своего пункта назначения. В состав всех современных операционных систем обязательно входит какая-нибудь версия этой команды. Синтаксис ее вызова выглядит довольно просто:

```
traceroute имя_компьютера
```

У этой команды очень много опций, большинство которых в повседневной работе не применяется. Как обычно, *имя_компьютера* может быть указано либо в виде DNS-имени, либо в виде IP-адреса. Вывод команды traceroute представляет собой просто перечень хостов, начинающийся с первого шлюза и заканчивающийся пунктом назначения.

Например, на нашем компьютере jaguar проверка хоста nubark дает следующие результаты:

```
$ traceroute nubark
traceroute to nubark (192.168.2.10), 30 hops max, 38 byte packets
1 lab-gw (172.16.8.254) 0.840 ms 0.693 ms 0.671 ms
2 dmz-gw (192.168.1.254) 4.642 ms 4.582 ms 4.674 ms
3 nubark (192.168.2.10) 7.959 ms 5.949 ms 5.908 ms
```

Эти результаты говорят о том, что для попадания с компьютера jaguar на хост nubark пакеты должны пройти два внутренних шлюза. Показано также полное время прохождения пакетов для каждого шлюза — проведено по три замера. Обычно количество переходов от одного хоста Internet к другому составляет более 15.

Команда traceroute работает путем записи искусственно заниженного значения в поле TTL (Time To Live — время жизни, или предельное число переходов) отправляемого пакета. Когда пакет приходит на очередной шлюз, его значение TTL уменьшается на единицу. Если шлюз обнаруживает, что значение TTL стало равным нулю, он удаляет пакет и возвращает хосту-отправителю специальное ICMP-сообщение: “Время жизни пакета истекло”.

Для трех самых первых пакетов команда `traceroute` задает значение TTL равным 1. Первый же шлюз, получивший пакет (в нашем примере это `lab-gw2`), обнаруживает, что его время жизни истекло. Тогда он удаляет пакет и посылает компьютеру `jaguar` указанное ICMP-сообщение, в поле отправителя которого записан IP-адрес шлюза. Команда `traceroute` обращается к службе DNS и по имеющемуся адресу находит имя шлюза.

📖 Об обратном поиске имен в DNS рассказывалось в разделе 15.7.

Для получения данных о следующем шлюзе посылается очередная группа пакетов, поле TTL которых равно 2. Первый шлюз маршрутизирует эти пакеты и уменьшает значение TTL на единицу. Второй шлюз удаляет пакеты и генерирует описанные выше ICMP-сообщения. Этот процесс продолжается до тех пор, пока пакеты не достигнут нужного компьютера; значение TTL при этом будет равно числу переходов.

Большинство маршрутизаторов посылает свои ICMP-сообщения через тот интерфейс, который является "ближайшим" к хосту-отправителю (т.е. имеет наименьшую метрику стоимости). Если, наоборот, запустить команду `traceroute` на хосте-получателе, чтобы узнать маршрут к исходному компьютеру, то, скорее всего, будет получен другой список IP-адресов, соответствующий тому же набору маршрутизаторов. Иногда весь набор шлюзов оказывается совершенно другим; такая конфигурация называется *асимметричной маршрутизацией*.

Для каждого значения TTL команда `traceroute` посылает три пакета, что иногда приводит к интересным результатам. Если промежуточный шлюз распределяет трафик по нескольким маршрутам, то пакеты могут возвращаться разными компьютерами. В таком случае команда `traceroute` сообщает обо всех полученных ответах.

Рассмотрим пример, в котором посредством команды `traceroute` определяется маршрут с компьютера в домене colorado.edu к домену xor.com:

```
rupertsberg$ traceroute xor.com
traceroute to xor.com (192.225.33.1), 30 hops max, 38 byte packets
 1  cs-gw3-faculty.cs.colorado.edu (128.138.236.3) 1.362 ms 2.144 ms  2.76 ms
 2  cs-gw-dmz.cs.colorado.edu (128.138.243.193)  2.720 ms 4.378 ms  5.052 ms
 3  engr-cs.Colorado.EDU (128.138.80.141)  5.587 ms  2.454 ms  2.773 ms
 4  hut-engr.Colorado.EDU (128.138.80.201)  2.743 ms  5.643 ms  2.772 ms
 5  cuatm-gw.Colorado.EDU (128.138.80.2)  5.587 ms  2.784 ms  2.777 ms
 6  204.131.62.6 (204.131.62.6)  5.585 ms  3.464 ms  2.761 ms
 7  border-from-BRAN.coop.net (199.45.134.81)  5.593 ms  6.433 ms  5.521 ms
 8  core-gw-eth-2-5.coop.net (199.45.137.14)  53.806 ms  *  19.202 ms
 9  xor.com (192.225.33.1)  16.838 ms  15.972 ms  11.204 ms
```

Мы видим, что, прежде чем покинуть домен `colorado.edu`, пакеты проходят пять внутренних шлюзов (от `cs-gw3-faculty` до `cuatm-gw`). На следующем переходе пакеты попадают в шлюз с адресом 204.131.62.6, который не имеет DNS-имени. Затем, после двух переходов в домене `coop.net`, пакеты оказываются в домене `xor.com`.

На восьмом переходе вместо одного из значений времени отображается звездочка. Это означает, что на один из отправленных пакетов не был получен ответ. Такая ситуация может быть вызвана перегрузкой сети, но это не единственное возможное объяснение. Команда `traceroute` использует низкоприоритетные ICMP-пакеты, которые отбрасываются многими маршрутизаторами, отдающими предпочтение "реальному" трафику. Появление нескольких звездочек в выводе команды `traceroute` не является причиной для беспокойства.

Если для какого-то шлюза отображаются все три звездочки, то это означает, что от него не было получено ни одного сообщения. Вероятнее всего, шлюз просто выключен. Правда, иногда шлюз или брандмауэр конфигурируют таким образом, чтобы он игнорировал устаревшие пакеты. В этом случае пакеты благополучно перейдут на следующий шлюз. Другой возможной причиной появления звездочек может быть то, что ответные пакеты от шлюза возвращаются слишком долго и команда `traceroute` перестает ожидать их прибытия.

Некоторые брандмауэры полностью блокируют передачу ICMP-сообщений о пакетах с истекшим временем жизни. Если маршрут пролегает через один из таких брандмауэров, команда `traceroute` не сможет получить информацию о шлюзах, находящихся за ним. Однако она узнает общее число переходов до пункта назначения, так как отправляемые эхо-пакеты пройдут весь маршрут. Некоторые брандмауэры могут также блокировать исходящие UDP-пакеты, которые команда `traceroute` посылает для инициирования ICMP-ответов. В этом случае команда не выдаст никакой полезной информации.

Медленная передача данных не всегда свидетельствует о неисправности сети. Некоторые сети действительно имеют большую задержку; беспроводные сети 802.11 являются хорошим тому примером. Медлительность может быть также следствием перегрузки сети, особенно если в ней используется технология CSMA/CD. В таких сетях (к ним относится Ethernet) осуществляются многократные попытки передачи пакета. Определить это можно по нелогичному значению полного времени прохождения пакета, так как коллизии увеличивают случайную составляющую поведения сети.

Иногда можно увидеть символы `!N` вместо значения времени. Это говорит о том, что соответствующий шлюз вернул сообщение о "недостижимости" сети, т.е. он не знает, куда посылать пакет. Сообщения о "недостижимости" хоста или протокола помечаются как `!H` или `!P`, соответственно. Шлюз, который возвращает любое из таких сообщений, обычно является последним в цепочке и, как правило, имеет проблемы с маршрутизацией (возможно, они вызваны разрывом связи): либо его статические маршруты заданы неверно, либо протоколы динамической маршрутизации не могут определить правильный маршрут для передачи пакета получателю.

Если команда `traceroute` не работает (или работает слишком медленно), это может быть вызвано превышением тайм-аута при попытках узнать имя компьютера в DNS. Возможно, на компьютере не функционирует служба DNS. В таком случае воспользуйтесь командой `traceroute -n`. Она будет выводить только IP-адреса шлюзов. Опция `-n` предотвращает использование службы DNS и может оказаться единственным способом заставить `traceroute` работать в поврежденной сети.

19.4. Команда `netstat`: получение информации о состоянии сети

Команда `netstat` собирает различную информацию о состоянии сетевого программного обеспечения, включая статистику сетевых интерфейсов, данные о маршрутизации и таблицы соединений. Никакого объединяющего звена во всех этих информационных блоках нет, просто они касаются функционирования сети. Воспринимайте команду `netstat` как "кухонную раковину" сетевых средств — она отображает различные данные о сети, которые просто больше никуда не вписываются. Здесь мы рассмотрим пять наиболее распространенных способов использования команды `netstat`, а это:

- анализ конфигурационных данных интерфейсов;
- мониторинг состояния сетевых соединений;
- выявление прослушивающих сетевых служб;
- изучение таблицы маршрутизации;
- просмотр статистических данных функционирования различных сетевых протоколов.

Анализ конфигурационных данных интерфейсов

Команда `netstat -i` отображает информацию о конфигурации и состоянии сетевых интерфейсов каждого из хостов. Она может служить неплохим средством для знакомства с сетевыми настройками нового компьютера. Для получения дополнительных деталей можно добавлять опцию `-e`.

```
$ netstat -i -e
Kernel Interface table
eth0   Link encap:Ethernet   HWaddr 00:02:B3:19:C8:82
       inet addr:192.168.2.1 Bcast:192.168.2.255 Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:1121527 errors:0 dropped:0 overruns:0 frame:0
       TX packets:1138477 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       Interrupt:7 Base address:0xef00

eth1   Link encap:Ethernet   HWaddr 00:02:B3:19:C6:86
       inet addr:192.168.1.13 Bcast:192.168.1.255 Mask:255.255.255.0
       UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
       RX packets:67543 errors:0 dropped:0 overruns:0 frame:0
       TX packets:69652 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:100
       Interrupt:5 Base address:0xed00

lo     Link encap:Local Loopback
       inet addr:127.0.0.1  Mask:255.0.0.0
       UP LOOPBACK RUNNING  MTU:3924  Metric:1
       RX packets:310572 errors:0 dropped:0 overruns:0 frame:0
       TX packets:310572 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0 txqueuelen:0
```

Данный компьютер имеет два сетевых интерфейса: один для обычного трафика и один для системного управления (`eth1`). Параметры `RX packets` и `TX packets` сообщают количество пакетов, принятых и переданных через каждый интерфейс с момента начальной загрузки системы. Далее сообщается о различных типах ошибок.

Количество ошибок должно составлять менее 1% от числа пакетов. Если частота появления ошибок высока, сравните эти показатели на нескольких соседних компьютерах. Большое число ошибок на одном компьютере свидетельствует о наличии проблемы в его интерфейсе или соединении, тогда как высокая частота ошибок, происходящих на всех компьютерах, скорее всего, обусловлена наличием проблемы в медиа-средствах или сети. Одной из наиболее распространенных причин высокой частоты ошибок является скорость Ethernet или двустороннее несоответствие, вызванное сбоем в системе автоматического зондирования или системе согласования взаимодействия.

Количество коллизий соответствует загруженности сети; наличие ошибок свидетельствует о проблемах с кабелями. Хотя коллизия является разновидностью ошибки, их число команда `netstat` подсчитывает отдельно. В поле `collisions` указывается число конфликтов, произошедших при отправке пакетов[2]. На основании этого числа можно вычислить долю ошибочных пакетов от общего количества отправленных пакетов (`TX packets`). "Нормальным" считается значение менее 3%, а значение выше 10% говорит о серьезной перегрузке сети. Коллизии никогда не происходят в двусторонних соединениях, которые функционируют должным образом.

Мониторинг состояния сетевых соединений

При отсутствии аргументов команда `netstat` выдает информацию о состоянии активных TCP- и UDP-портов. Неактивные ("прослушивающие") серверы, ожидающие поступления запросов на установление соединений, как правило, не отображаются; о них можно узнать с помощью команды `netstat -a` [3]. Вывод будет выглядеть примерно так:

```
$ netstat -a
Active Internet connections (servers and established)
Proto      Recv-Q   Send-Q   Local Address  Foreign Address    State
tcp        0        0        *:ldap         *:*                LISTEN
tcp        0        0        *:mysql        *:*                LISTEN
tcp        0        0        *:imaps        *:*                LISTEN
tcp        0        0        bull:ssh       dhcp-32hw:4208     ESTABLISHED
tcp        0        0        bull:imaps     nubark:54195       ESTABLISHED
tcp        0        0        bull:http      dhcp-30hw:2563     ESTABLISHED
tcp        0        0        bull:imaps     dhcp-18hw:2851     ESTABLISHED
tcp        0        0        *:http         *:*                LISTEN
tcp        0        0        bull:37203     baikal:mysql       ESTABLISHED
tcp        0        0        *:ssh          *:*                LISTEN…
...
```

Этот пример был взят с компьютера `otter` и сильно усечен; например соединения с сокетами UDP и UNIX не отображаются. Показанный выше вывод свидетельствует о наличии одного входящего соединения по протоколу SSH, двух входящих IMAPS-соединений, одного входящего HTTP-соединения, одного исходящего соединения с MySQL и группы портов, ожидающих других запросов на подключение.

Адреса представлены в формате *имя_компьютера.служба*, где *служба* — это номер порта. Для известных служб порты указаны в символическом виде (соответствия между номерами портов и их именами определены в файле `/etc/services`). Получить адреса и порты в числовом виде можно при помощи опции `-n`. Как и в случае большинства сетевых отладочных средств, если служба DNS не работает, без флага `-n` команда `netstat` будет выполняться очень медленно.

В колонках `Send-Q` и `Recv-Q` показывается, сколько запросов находится во входящих и исходящих очередях на данном компьютере. На другом конце соединения размеры очередей могут быть иными. Желательно, чтобы эти значения были близки к нулю и не были ненулевыми постоянно. Конечно, если команда `netstat` запускается через сетевой терминал, для ее соединения размер исходящей очереди, скорее всего, никогда не будет равен нулю.

---

[2] Это поле имеет смысл только в сетях с технологией CSMA/CD, например Ethernet.

[3] Соединения с именованными UNIX-сокетами тоже отображаются, но поскольку они не имеют отношения к сетевым соединениям, они здесь не рассматриваются.

Состояние соединения определено только для протокола TCP. Протокол UDP не проверяет факт установления соединения. Наиболее распространенные состояния таковы: ESTABLISHED (установлено) — для активных соединений, LISTENING (ожидание) — для серверов, ожидающих поступления запросов (при отсутствии опции -a обычно не показываются), TIME_WAIT (ожидание закрытия) — для соединений, находящихся в процессе закрытия.

Эта информация полезна главным образом для устранения проблем на высоком уровне, если, конечно, базовые сетевые средства работают нормально. Команда netstat позволяет проверить правильность настройки серверов и диагностировать определенные виды нарушений связи, особенно при работе с протоколом TCP. Например, если соединение находится в состоянии SYN_SENT, то это означает наличие процесса, который пытается установить контакт с несуществующим или недоступным сервером.

Если команда netstat сообщает о большом количестве соединений, находящихся в состоянии SYN_WAIT, то компьютер, очевидно, не в состоянии обработать имеющееся число запросов на установление соединений. Такая ситуация может быть вызвана ограничениями системного ядра или даже злонамеренными попытками вызвать перегрузку.

📖 О настройке ядра более подробно будет рассказываться в главе 28.

Выявление прослушивающих сетевых служб

В нашу обеспокоенную безопасностью эру часто можно услышать вопрос: “Какие процессы на этой машине прослушивают сеть, ожидая входящих соединений?”. Команда netstat -a отображает все порты, которые находятся в активном состоянии и занимаются прослушиванием (т.е. все TCP-порты в состоянии LISTEN и, потенциально, все UDP-порты), но на загруженном компьютере они могут теряться в шуме установленных TCP-соединений. Поэтому, чтобы просмотреть только прослушивающие порты, лучше использовать команду netstat -l. Формат вывода этой команды такой же, как у команды netstat-a.

Если добавить флаг -p, команда netstat также отобразит конкретные процессы, ассоциируемые с каждым прослушивающим портом. В показанном ниже в качестве примера выводе отображаются три общих службы (sshd, sendmail и named), за которыми идет одна необычная:

```
$ netstat -lp
...
tcp        0    0    0.0.0.0:22    0.0.0.0:*     LISTEN    23858/sshd
tcp        0    0    0.0.0.0:25    0.0.0.0:*     LISTEN    10342/sendmail
udp        0    0    0.0.0.0:53    0.0.0.0:*               30016/named
udp        0    0    0.0.0.0:962   0.0.0.0:*               38221/mudd
...
```

Здесь служба mudd с PID-номером 38221 прослушивает UPD-порт под номером 962. В зависимости от существующей политики в отношении устанавливаемого пользователями ПО, возможно с ней нужно будет разобраться.

Изучение таблицы маршрутизации

Команда netstat -r отображает таблицу маршрутизации ядра. Следующие результаты были получены на компьютере, который имеет два сетевых интерфейса (в разных дистрибутивах Linux вывод этой команды может немного отличаться):

```
$ netstat -rn
Kernel IP routing table
Destination     Gateway       Genmask       Flags  MSS Window  irtt Iface
192.168.1.0     0.0.0.0       255.255.255.0 U      0   0       0    eth0
10.2.5.0        0 0.0.0.0     255.255.255.0 U      0   0       0    eth1
127.0.0.0       0.0.0.0       255.0.0.0     U      0   0       0    lo
0.0.0.0         192.168.1.254 0.0.0.0       UG     0   0       40   eth0
...
```

Пункты назначения и шлюзы могут быть представлены доменными именами либо IP-адресами. Флаг `-n` задает вывод IP-адресов.

В колонке `Flags` отображаются флаги, характеризующие маршрут: `U` (up) — активный, `G` (gateway) — шлюзовой, `H` (host) — хостовой (связан с конкретным адресом, а не сетью). Флаги `U`, `G` и `H`, стоящие вместе, обозначают маршрут к компьютеру, проходящий через промежуточный шлюз. Флаг `D` (не показан) обозначает маршрут, полученный в результате переадресации по протоколу ICMP. Остальные поля содержат статистические данные о маршруте: текущее число TCP-соединений по этому маршруту, количество отправленных пакетов и имя используемого интерфейса.

📖 О таблицах маршрутизации рассказывалось в начале раздела 12.5.

Рассмотренный вариант команды `netstat` полезен для проверки правильности таблицы маршрутизации. Особенно важно убедиться в наличии и корректности стандартного маршрута. Он обозначается в виде адреса со всеми нулями (0.0.0.0). Отсутствие стандартного маршрута возможно, но такая конфигурация крайне нетипична.

Просмотр статистических данных функционирования различных сетевых протоколов

Команда `netstat -s` отображает содержимое всевозможных счетчиков, используемых в сетевых программах. Информация разбивается на разделы в соответствии с протоколами: IP, ICMP, TCP и UDP. Ниже показаны отдельные фрагменты листинга команды `netstat -s`, полученного на типичном компьютере; они отредактированы, чтобы остались только наиболее интересные данные.

```
Ip:
  671349985 total packets received (всего полученных пакетов)
  0 forwarded (переслано)
  345 incoming packets discarded (345 отклоненных входящих пакетов)
  667912993 incoming packets delivered (667912993 доставленных входящих пакетов)
  589623972 requests sent out (589623972 разосланных запросов)
  60 dropped because of missing route (60 пропущенных из-за отсутствия маршрута)
  203 fragments dropped after timeout (203 фрагмента, пропущенных после тайм-аута)
```

Обязательно нужно следить за количеством пропущенных и отклоненных пакетов. Отклонение нескольких входящих пакетов допустимо, но резкое увеличение этого показателя обычно свидетельствует о нехватке памяти или каких-нибудь других проблемах с ресурсами.

```
Icmp:
  242023 ICMP messages received (242023 полученных ICMP-сообщений)
  912 input ICMP message failed. (912 не полученных ICMP-сообщений)
  ICMP input histogram: (гистограмма входных данных ICMP)
    destination unreachable: 72120 (не удалось достичь пункта назначения)
```

```
    timeout in transit: 573 (тайм-аут во время передачи)
    echo requests: 17135 (эхо-запросы)
    echo replies: 152195 (эхо-ответы)
  66049 ICMP messages sent (66049 отправленных ICMP-сообщений)
  0 ICMP messages failed (0 неотправленных ICMP-сообщений)
  ICMP output histogram: (гистограмма выходных данных ICMP)
    destination unreachable: 48914 (не удалось достичь пункта назначения)
    echo replies: 17135 (эхо-ответы)
```

В этом примере количество эхо-запросов совпадает с количеством эхо-ответов. Обратите внимание на то, что сообщения "destination unreachable" (не удалось достичь пункта назначения) могут генерироваться даже тогда, когда все пакеты, казалось бы, маршрутизируются нормально. Например, дефектные пакеты рано или поздно достигают шлюза, который отклоняет их, и тогда сообщения об ошибке отправляются обратно по цепочке шлюзов.

```
Tcp:
4442780 active connections openings (4442780 активных открытий соединений)
1023086 passive connection openings (1023086 пассивных открытий соединений)
50399 failed connection attempts (50399 неудачных попыток установки соединения)
0 connection resets received (0 запросов на повторную установку соединения получено)
44 connections established (44 соединения установлено)
666674854 segments received (666674854 сегментов получено)
585111784 segments send out (585111784 сегментов отправлено)
107368 segments retransmitted (107368 сегментов передано повторно)
86 bad segments received. (86 дефектных сегментов получено)
3047240 resets sent (3047240 запросов на повторную установку соединения отправлено)

Udp:
4395827 packets received (4395827 пакетов получено)
31586 packets to unknown port received. (Получено 31586 пакетов,
                                         отправленных на неизвестный порт)
0 packet receive errors (0 ошибок при получении пакетов)
4289260 packets sent (4289260 пакетов отправлено)
```

Рекомендуем определить приемлемые диапазоны этих показателей, чтобы можно было сразу же распознавать ненормальное состояние сети.

19.5. SAR: анализ текущего состояния интерфейсов

Один из наилучших способов идентифицировать проблемы в сети — это взять и посмотреть, что же происходит в сети прямо сейчас. Какое количество пакетов было отправлено за последние пять минут через данный интерфейс? Какое количество байтов они занимали? Не было ли каких-нибудь коллизий или других ошибок? На все эти вопросы можно ответить, понаблюдав за текущим функционированием интерфейса.

На традиционных системах UNIX для этой цели используется команда `netstat -i`. К сожалению, в системах Linux функция этой команды, отвечающая за предоставление отчетов о текущем функционировании интерфейса, работает плохо. Поэтому мы рекомендуем пользоваться совершенно другой командой — командой `sar`. (О возможностях этой команды с точки зрения общего системного мониторинга более подробно будет рассказываться в разделе 25.3.) В большинстве дистрибутивов утилита `sar` по умолчанию не устанавливается, но, как правило, всегда доступна в виде дополнительного пакета.

Чтобы заставить `sar` генерировать отчет о функционировании интерфейсов каждые две секунды на протяжении одной минуты (т.е. сгенерировать всего 30 отчетов), нужно использовать такой синтаксис: `sar -n DEV 2 30`. Аргумент `DEV` — это литеральное ключевое слово, а не указатель места ввода имени устройства или интерфейса.

Вывод этой утилиты состоит из мгновенно считываемых усредненных данных об использовании сетевого интерфейса в виде байтов и пакетов. Показанный ниже пример вывода был получен на компьютере Red Hat с двумя физическими интерфейсами. Очевидно, что второй физический интерфейс (`eth1`) не используется.

```
17:50:43  IFACE  rxpck/s  txpck/s  rxbyt/s txbyt/s  rxcmp/s  txcmp/s rxmcst/s
17:50:45     lo     3.61     3.61   263.40  263.40     0.00     0.00      0.00
17:50:45   eth0    18.56    11.86 1364.43 1494.33     0.00     0.00      0.52
17:50:45   eth1     0.00     0.00     0.00    0.00     0.00     0.00      0.00
```

В первых двух колонках отображается время сбора данных и имена сетевых интерфейсов. В следующих двух колонках отображается количество полученных и переданных пакетов (соответственно). Колонки `rxbyt/s` и `txbyt/s`, пожалуй, являются наиболее полезными, поскольку в них отображается фактическое количество полученных и переданных байтов. В трех последних колонках содержатся статистические данные о сжатых (`rxcmp/s`, `txcmp/s`) и многоадресных пакетах (`rxmcst/s`).

Команда `sar-n DEV` особенно удобна для отслеживания источника ошибок. `ifconfig` может уведомлять о существовании проблем, но не позволяет понять, являются ли ошибки следствием продолжительной низкоуровневой проблемы или появились в результате краткого, но крайне негативного события. Разобраться в том, что происходит, можно лишь понаблюдав за поведением сети при различных условиях нагрузки на протяжении какого-то времени. Проверяя вывод `sar-n DEV`, попробуйте выполнить команду `ping` с большим значением для размера пакета.

19.6. Анализаторы пакетов

Программы `tcpdump` и Wireshark относятся к классу утилит, известных как *анализаторы пакетов*. Они следят за трафиком в сети и регистрируют либо выводят на экран пакеты, удовлетворяющие заданным критериям. Например, можно перехватывать все пакеты, посылаемые на какой-то компьютер или с него, либо TCP-пакеты, относящиеся к конкретному сетевому соединению.

Анализаторы пакетов полезны как для решения известных проблем, так и для выявления абсолютно новых. Важно время от времени запускать эти программы и проверять, нормально ли обрабатывается сетевой трафик.

Поскольку анализаторы должны уметь перехватывать пакеты, которые локальный компьютер обычно не получает (или на которые не обращает внимания), базовые сетевые аппаратные средства должны разрешать доступ к каждому пакету. Это характерно для широковещательных технологий, в частности Ethernet, так же как и для большинства других современных локальных сетей.

Анализаторы пакетов должны иметь доступ к низкоуровневому трафику, поэтому их работе могут мешать коммутаторы, одной из функций которых является препятствование распространению “ненужных” пакетов. Однако с помощью анализаторов удается получить полезную информацию даже в коммутируемых сетях. Можно, к примеру, обнаружить проблемы, затрагивающие широковещательные и групповые пакеты. Объем выдаваемой информации зависит от модели коммутатора.

О коммутаторах рассказывалось в разделе 14.2.

Аппаратный интерфейс должен не только иметь возможность получать доступ ко всем сетевым пакетам, но также передавать эти пакеты вверх на программный уровень. Ведь обычно адреса пакетов проверяются на аппаратном уровне, а ядру предоставляются лишь широковещательные/групповые пакеты и те, которые адресованы данному компьютеру. В *беспорядочном режиме* (promiscuous mode) интерфейс позволяет ядру получать все сетевые пакеты, даже если они предназначены для других компьютеров.

Анализаторы понимают многие форматы пакетов, используемые стандартными сетевыми службами, и часто могут отображать содержимое пакетов в удобном для восприятия виде. Это облегчает пользователю анализ сеанса между двумя программами. Существуют анализаторы, которые кроме заголовка пакета выводят и его содержимое в текстовом виде, что полезно при исследовании протоколов верхних уровней. Некоторые из этих протоколов пересылают информацию (в том числе и пароли) по сети в незашифрованном виде, поэтому следует заботиться о сохранении конфиденциальности пользовательских данных.

Каждый из рассматриваемых нами дистрибутивов поставляется с каким-нибудь анализатором пакетов. В связи с тем, что анализатору необходим доступ к пакетам на самом низком уровне, он должен запускаться от имени пользователя root. Подобное ограничение снижает шансы обычных пользователей получить доступ ко всему сетевому трафику, но на самом деле этот барьер можно преодолеть. Во многих организациях анализаторы пакетов удалены с большинства компьютеров с целью уменьшения риска злоупотребления этими программами. Если такая мера невозможна, следует проверять интерфейсы системы, чтобы они не работали в “беспорядочном” режиме без ведома или согласия администратора. В Linux интерфейс, работающий в “беспорядочном” режиме, помечается в выводе команды `ifconfig` флагом `PROMISC`. Выявить наличие в сети таких интерфейсов можно с помощью специальных программ, например PromiScan (`www.securityfriday.com`).

Утилита `tcpdump`: самый лучший анализатор

Замечательный анализатор пакетов `tcpdump`, написанный Ваном Якобсоном, входит в состав большинства дистрибутивов Linux. Он уже давно считается промышленным стандартом, и другие анализаторы читают и записывают файлы трассировки в формате `tcpdump`.

По умолчанию утилита `tcpdump` использует первый найденный ею сетевой интерфейс. Если она выбрала не тот интерфейс, то посредством опции `-i` следует задать нужный. В случае неисправности службы DNS или если необходимо видеть адреса компьютеров, воспользуйтесь опцией `-n`. Эта опция имеет большое значение, так как медленная работа DNS может привести к удалению пакетов еще до того, как они будут проанализированы утилитой `tcpdump`. Опция `-v` позволяет получить более детальное описание каждого пакета, а самые подробные результаты выдаются при задании опции `-vv`. Если указана опция `-w`, утилита сохраняет перехваченные пакеты в файле. Для чтения пакетов из файла предназначена опция `-r`.

Например, ниже показана усеченная версия вывода, полученного на компьютере `nubark`. Спецификация фильтра `host bull` ограничивает отображаемые пакеты только теми, в которых непосредственно используется `bull` либо как источник, либо как пункт назначения.

```
# sudo tcpdump host bull
12:35:23.519339 bull.41537 > nubark.domain: A? atrust.com. (28) (DF)
12:35:23.519961 nubark.domain > bull.41537: A 66.77.122.161 (112) (DF)
```

В первой строке содержится информация о том, что с компьютера `bull` на компьютер `nubark` был отправлен пакет с запросом к службе DNS по поводу atrust.com. Во второй строке приводятся сведения о втором пакете: он представляет собой IP-адрес ассоциируемого с этим именем компьютера, который выглядит так: 66.77.122.161. Обратите внимание на отображаемую слева временную метку и то, как `tcpdump` воспринимает протокол уровня приложений (каковым в данном случае является DNS). Номер порта на компьютере `bull` является произвольным и отображается в числовом формате (41537), но поскольку номер порта сервера (53) хорошо известен, `tcpdump` вместо этого отображает его символическое имя (“domain”).

Анализаторы пакетов могут генерировать огромное количество информации — огромное не только для вас, но и также для лежащей в основе операционной системы. Во избежание это проблемы в загруженных сетях `tcpdump` позволяет задавать довольно сложные фильтры. Например, следующий фильтр собирает только входящий Web-трафик из конкретной подсети:

```
# sudo tcpdump src net 192.168.1.0/24 and dst port 80
```

На `man`-странице утилиты `tcpdump` приведен ряд хороших примеров различных фильтров, а также полный список ключевых слов, которые могут использоваться в командной строке[4].

Wireshark: визуальный анализатор пакетов

Для любителей анализировать пакеты в графическом режиме можем порекомендовать программу Wireshark. Она доступна на Web-сайте `www.wireshark.com` и распространяется на условиях общей открытой лицензии GNU (GPL). Программа представляет собой визуальный анализатор пакетов формата GTK+ (инструментальный набор GIMP), обладающий гораздо большим числом функций, чем многие коммерческие аналоги. Существуют версии программы для Linux и Windows.

Помимо анализа пакетов программа Wireshark располагает рядом дополнительных полезных возможностей. В частности, она может читать и записывать файлы трассировки множества форматов, включая:

- TCPDUMP;
- NAI Sniffer;
- Sniffer Pro;
- NetXray;
- Snoop;
- Shomiti Surveyor;
- Microsoft Network Monitor;
- Novell LANalyzer;
- Cisco Secure IDS iplog;

[4] Если для фильтров требуются большие возможности, чем имеющиеся у `tcpdump`, можно попробовать использовать утилиту `ngrep`, которая позволяет фильтровать пакеты в соответствии с их содержимым.

Другое преимущество программы заключается в том, что можно щелкнуть на изображении пакета в TCP-потоке и попросить программу "воссоздать" (соединить) полезное содержимое всех пакетов потока. Это позволяет быстро оценить объем данных, передаваемых в рамках TCP-диалога, например при передаче почтового сообщения по сети[5].

У Wireshark имеются фильтры захвата, которые функционируют точно так же, как и аналогичные фильтры утилиты `tcpdump`. Однако будьте осторожны: у Wireshark также имеются фильтры отображения, которые больше влияют на отображаемые данные, чем на те, что фактически перехватывает анализатор. Как ни странно, синтаксис фильтров отображения сильно отличается от синтаксиса фильтров захвата.

Утилита Wireshark является чрезвычайно мощным средством анализа и входит в состав набора инструментов практически каждого эксперта по сетям. Более того, она также является бесценным помощником для тех, кто только начинает изучать пакетные сети. В справочном меню Wireshark доступно множество замечательных примеров, с которых можно начать. Не бойтесь экспериментировать!

19.7. Протоколы управления сетями

Усложнение и разрастание сетей вызвало потребность в разработке средств управления сетями. Поставщики операционных систем и разработчики стандартов подходили к решению этой задачи самыми разными способами. Наиболее значительным результатом стало появление ряда стандартных протоколов управления сетевыми устройствами и множества высокоуровневых программных средств, в которых эти протоколы используются.

Протоколы управления сетями определяют стандартные методы зондирования какого-либо устройства с целью выявления его сетевых соединений, конфигурации и общего состояния. Кроме того, протоколы позволяют модифицировать часть этой информации, чтобы можно было стандартизировать управление различными устройствами на сетевом уровне и осуществлять это управление с центральной станции.

Самым распространенным протоколом управления, используемым в сетях TCP/IP, является SNMP (Simple Network Management Protocol — простой протокол управления сетями). Несмотря на название, этот протокол достаточно сложен. Он определяет иерархическое пространство имен для объектов управления и методы чтения/записи данных, относящихся к этим объектам, на каждом хосте иерархии. Он также определяет способ, которым управляемые серверы и устройства ("агенты") посылают сообщения о происходящих событиях ("прерывания") станциям управления.

Протокол SNMP сам по себе является простым; самая сложная его часть находится над протокольным уровнем и заключается в соглашениях по организации пространства имен и в излишне вычурном словаре, который окружает SNMP как защитная оболочка. Пока не задумываешься слишком глубоко о внутренних механизмах протокола SNMP, его использование не представляет сложности.

Существует и ряд других стандартов управления сетями. Многие из них созданы организацией DMTF (Distributed Management Task Force — рабочая группа по разработке распределенных систем управления), которая реализовала такие концепции, как WBEM (Web-Based Enterprise Management — система управления предприятием на основе Web-технологий), DMI (Desktop Management Interface — интерфейс управления ком-

[5] То же самое можно сделать в режиме командной строки с помощью утилиты `tcpflow`, передав ей результаты работы команды `tcpdump`.

пьютером) и CIM (Conceptual Interface Model — концептуальная модель интерфейса). Некоторые из этих концепций, в частности DMI, приняты рядом известных фирм-производителей и могут служить полезным дополнением (а иногда и заменой) протоколу SNMP. Запатентованных протоколов управления в настоящее время тоже существует очень много. Однако пока основным средством управления сетями и системами Linux остается все же SNMP.

Поскольку SNMP — абстрактный протокол, для его использования нужны программа-сервер ("агент") и программа-клиент ("менеджер"). (Как ни странно, серверная сторона SNMP является управляемым элементом, а клиентская — управляющим.) Клиентом может быть как простая утилита, работающая в режиме командной строки, так и выделенная станция управления, на мониторе которой в графическом виде отображается сеть вместе со всеми неполадками.

Выделенные станции управления сетью являются главной причиной существования самих протоколов управления. Большинство программных продуктов позволяет строить не только логическую, но и топографическую модель сети. Обе модели выводятся на экран, при этом постоянно отображается текущее состояние каждого сетевого компонента.

Подобно тому, как график может показать скрытый смысл, заложенный в заполненной числами странице, так и станция управления сетью способна обобщить и представить состояние крупной сети в форме, приемлемой для человека. Другим способом такую информационную сводку получить практически невозможно.

Основное преимущество протокола SNMP в том, что абсолютно все типы сетевых аппаратных средств выводятся на один уровень. Linux-системы в основном похожи друг на друга, чего нельзя сказать о маршрутизаторах, шлюзах и остальных низкоуровневых компонентах. При использовании протокола SNMP все они начинают "общаться" на одном языке и могут зондироваться, сбрасываться в начальное состояние и конфигурироваться с центральной станции. Очень удобно, когда есть один согласованный интерфейс, применимый ко всем сетевым устройствам.

19.8. SNMP: простой протокол управления сетями

Когда в начале 90-х гг. протокол SNMP впервые появился на рынке, возникла своеобразная "золотая лихорадка". Сотни компаний выпустили собственные пакеты управления по протоколу SNMP. Кроме того, многие поставщики аппаратных и программных средств стали включать в свои продукты SNMP-агенты.

Прежде чем погрузиться в дебри SNMP, заметим, что терминология, употребляемая в этой области, одна из самых бестолковых и невразумительных. Стандартные названия объектов и концепций SNMP будут активно уводить читателей от понимания их назначения, так что запаситесь терпением.

Структура протокола SNMP

В SNMP данные организованы иерархически, причем структура иерархии жестко определена. Это делает пространство имен универсальным и расширяемым, по крайней мере теоретически. Большие его области оставлены для перспективного использования; дополнения, создаваемые поставщиками операционных систем, локализуются в определенных диапазонах во избежание конфликтов. Для формирования пространства имен используются так называемые *базы управляющей информации* (Management Information

Base, MIB). Это структурированные текстовые файлы, которые содержат описания данных, доступных по протоколу SNMP. Ссылки на конкретные переменные, описываемые в базе данных, называются *идентификаторами объектов* (Object Identifier, OID).

В переводе на человеческий язык это означает всего лишь, что протокол SNMP определяет иерархическое пространство имен переменных, хранящих "интересные" параметры системы. Идентификатор объекта (OID) — это просто модный способ именования конкретного управляемого блока информации.

Иерархия SNMP напоминает иерархию имен файловой системы. В качестве символа-разделителя используется точка, а каждому хосту иерархии присваивается не имя, а номер. Для облегчения ссылок хостам присваиваются также текстовые имена, но схема именования выходит за рамки самой иерархии и определяется на высоком уровне (это похоже на связь между именами компьютеров и их IP-адресами).

К примеру, идентификатор OID, соответствующий показателю общего времени работы системы, выглядит так: 1.3.6.1.2.1.1.3. В текстовом виде его можно записать следующим образом:

```
iso.org.dod.internet.mgmt.mib-2.system.sysUpTime
```

Верхние уровни иерархии SNMP носят искусственный характер и обычно не содержат полезных данных. По сути, интересная информация начинает появляться только на уровне `iso.org.dod.internet.mgmt` (в числовом виде — 1.3.6.1.2).

Основная административная база данных SNMP для протоколов TCP/IP (MIB-I) обеспечивает доступ к наиболее важным управляющим данным: информации о системе, ее интерфейсах, преобразовании адресов и протоколах (IP, ICMP, TCP, UDP и других). В документе RFC1213 описана новая, более полная версия этой базы: MIB-II. Большинство SNMP-серверов поддерживает базу MIB-II. В табл. 19.1 приведена выборка хостов иерархии из пространства имен MIB-II.

Таблица 19.1. Некоторые идентификаторы из базы MIB-II

OID[a]	Тип	Содержание
`system.sysDescr`	строка	Информация о системе: производитель, модель, тип ОС и т.д.
`system.sysLocation`	строка	Физическое местонахождение компьютера
`system.sysContact`	строка	Информация о владельце компьютера
`system.sysName`	строка	Имя системы (обычно это полное DNS-имя)
`interfaces.ifNumber`	целое	Количество имеющихся сетевых интерфейсов
`interfaces.ifTable`	таблица	Таблица с информацией о каждом интерфейсе
`ip.ipForwarding`	целое	1, если система является шлюзом, иначе 2
`ip.ipAddrTable`	таблица	Таблица данных IP-адресации (маски и т.д.)
`ip.ipRouteTable`	таблица	Системная таблица маршрутизации
`icmp.icmplnRedirects`	целое	Число полученных директив переадресации протокола ICMP
`icmp.icmplnEchos`	целое	Число полученных эхо-пакетов
`tcp.tcpConnTable`	таблица	Таблица текущих TCP-соединений
`udp.udpTable`	таблица	Таблица с информацией о UDP-сокетах, через которые серверы ожидают прием запросов

[a] Относительно хоста `iso.org.dod.internet.mgmt.mib-2`.

Помимо основной административной базы данных существуют базы для различных типов аппаратных интерфейсов и протоколов. Имеются также базы данных по отдельным поставщикам и конкретным изделиям.

База MIB — всего лишь соглашение об именовании управляющих данных. Для того чтобы эта схема заработала, ее необходимо подкрепить программой-агентом, которая будет обеспечивать соответствие содержимого SNMP-переменных текущему состоянию устройства. Код для основной базы MIB (в настоящее время MIB-II) поставляется с большинством SNMP-агентов Linux. Некоторые агенты разрешают подключать дополнительные базы данных.

Операции протокола SNMP

В пространстве имен SNMP определено всего четыре базовые операции: `get` (прочитать), `get-next` (прочитать следующий), `set` (записать) и `trap` (прерывание).

Операции `get` и `set` — это базовые операции чтения и записи данных на узле иерархии, который определяется конкретным значением OID. Операция `get-next` используется для последовательного прохода по базе MIB, а также для чтения содержимого таблиц.

Прерывание (операция `trap`) — это неожиданное уведомление клиента о том, что на сервере произошло нестандартное событие. Определен ряд стандартных прерываний, в том числе уведомления вида "я только что включился", отчеты о сбое или восстановлении сетевого соединения и сообщения о различных проблемах с маршрутизацией и аутентификацией. Широко применяются и нестандартные прерывания, например для отслеживания значений требуемых SNMP-переменных. Если эти значения выходят за границы установленного диапазона, выдается соответствующее сообщение. Способ определения получателей таких сообщений зависит от реализации агента.

Поскольку сообщения SNMP потенциально могут изменять информацию о конфигурации компьютера, необходим какой-то механизм защиты. Простейшая версия защиты SNMP основывается на концепции *групповой строки* (community string), которая представляет собой просто очень завуалированный синоним слова "пароль". Обычно существует одна групповая строка для получения доступа для чтения и еще одна для выполнения записи.

Хотя многие организации до сих пор пользуются исходной, основанной на групповых строках аутентификацией, версия 3 протокола SNMP уже предлагает методы управления доступом с более высокой степенью защиты. Конфигурирование такого более совершенного механизма защиты требует приложения дополнительных усилий, но сокращение степени риска того стоит. Если по какой-то причине использовать версию 3 не представляется возможным, следует по крайней мере убедиться в том, что выбрана трудно разгадываемая групповая строка (hard-to-guess string).

RMON: база MIB для дистанционного мониторинга

База RMON (remote monitoring — дистанционный мониторинг) накапливает данные об общих характеристиках сети (т.е. тех, которые не относятся к конкретному устройству). Сетевые анализаторы, или "зонды", могут собирать информацию о загруженности и производительности сети. Полезные данные группируются, подвергаются статистической обработке и доставляются на центральную станцию управления для анализа и графического воспроизведения. Многие зонды имеют буферы для перехваченных пакетов и могут работать подобно утилите `tcpdump`.

База RMON формально описана в документе RFC1757, который был принят в качестве чернового стандарта в 1995 году. Она разделяется на девять "групп RMON". Каждая группа хранит собственный набор статистических данных. Если сеть достаточно велика и имеет много глобальных соединений, необходимо рассмотреть возможность приобретения зондов для снижения SNMP-трафика через глобальные соединения. При наличии итоговых статистических данных отпадет необходимость в дистанционном сборе первичных данных. Многие коммутаторы и маршрутизаторы поддерживают базы RMON и хранят в них собственные статистические данные.

19.9. Агент NET-SNMP

После публикации первой версии стандарта SNMP в университете Карнеги-Меллона и Массачусетсском технологическом институте были разработаны реализации протокола. Университетская реализация (CMU SNMP) была более полной и быстро стала стандартом де-факто. После прекращения разработки SNMP-продуктов в университете Карнеги-Меллона работу продолжили специалисты Калифорнийского университета в Дэвисе. Когда код системы стабилизировался, она была передана для дальнейшего сопровождения компании SourceForge. В настоящее время пакет называется NET-SNMP.

Этот пакет теперь считается официальной бесплатной реализацией протокола SNMP для Linux. На самом деле сегодня многие производители сетевых устройств интегрируют его в свои продукты. В его состав входят: SNMP-агент, несколько утилит командной строки, сервер для получения сообщений о прерываниях (`trap`) и даже библиотека для разработки SNMP-приложений. Ниже мы рассмотрим работу самого агента, а об утилитах поговорим чуть позже. Последнюю версию пакета можно получить на Web-хосте `net-snmp.sourceforge.net`.

Как и в других реализациях протокола, агент NET-SNMP собирает данные о локальном компьютере и предоставляет их SNMP-менеджерам по сети. По умолчанию инсталлируются базы MIB, содержащие статистические данные об использовании сетевых интерфейсов, памяти, дисков и центрального процессора. Возможности агента достаточно велики, так как он способен запустить любую Linux-команду и представить ее выходные данные в качестве SNMP-ответа. Это позволяет посредством протокола SNMP осуществлять мониторинг событий, происходящих в системе.

По умолчанию агент инсталлируется под именем `/usr/sbin/snmpd`. Обычно он запускается на этапе начальной загрузки системы и считывает параметры конфигурации из файлов, находящихся в каталоге `/etc/snmp`. Наиболее важный из этих файлов называется `snmpd.conf`; он содержит большинство конфигурационных параметров и описание множества доступных методов сбора данных. При разработке пакета его авторы полагали, что пользователи будут редактировать только файл `snmpd.local.conf`. Но на практике приходится хотя бы один раз изменять и файл `snmpd.conf`, чтобы отключить те методы сбора данных, использование которых не планируется.

Сценарий `configure` пакета NET-SNMP позволяет задать используемый по умолчанию журнальный файл и ряд других локальных параметров. С помощью опции `-l`, указываемой при вызове демона `snmpd`, можно выбрать другой журнальный файл, а благодаря опции `-s` журнальные сообщения направляются в систему Syslog. Перечень наиболее важных опций демона `snmpd` приведен в табл. 19.2. Мы рекомендуем всегда задавать опцию `-a`. При поиске неисправностей удобно применять опции `-V`, `-d` и `-D`, которые позволяют получать более детальную информацию о происходящих в системе событиях.

Таблица 19.2. Опции демона `snmpd` пакета NET-SNMP

Опция	Назначение
`-l` *файл*	Направление журнальной информации в указанный файл
`-a`	Регистрация адресов всех SNMP-соединений
`-d`	Регистрация содержимого каждого SNMP-пакета
`-V`	Включение режима подробного описания событий
`-D`	Регистрация отладочной информации
`-h`	Отображение аргументов демона `snmpd`
`-H`	Отображение директив конфигурационного файла
`-A`	Добавление данных в журнальный файл, а не его перезапись
`-s`	Направление журнальной информации в систему Syslog

Следует заметить, что существует большое количество полезных модулей Perl для SNMP. Тем, кто собирается писать собственные сценарии управления сетью, рекомендуем поискать примеры в архиве CPAN[6].

19.10. Программы управления сетями

Начнем мы этот раздел с рассмотрения самых простых средств управления SNMP, а именно — команд, поставляемых вместе с пакетом NET-SNMP. Эти команды могут помочь познакомиться с SNMP, а также прекрасно подходят для одноразовых проверок значений конкретных идентификаторов OID. Затем мы рассмотрим программу Cacti, которая умеет генерировать замечательные хронологические графики изменения SNMP-значений, и программу Nagios, которая представляет собой хорошую систему мониторинга событий. В завершение мы приведем несколько рекомендаций касательно того, на что следует обращать внимание при покупке коммерческих систем.

Команды пакета NET-SNMP

Даже если Linux-система поставляется с собственным SNMP-сервером, для полноценного администрирования все равно может понадобиться скомпилировать и инсталлировать какие-нибудь клиентские команды из пакета NET-SNMP. В табл. 19.3 перечислены наиболее часто используемые команды этого пакета.

Перечисленные команды чрезвычайно удобно использовать в сценариях. Например, часто требуется сценарий, в котором данные, полученные командой `snmpget`, каждые несколько минут сохраняются в текстовом файле. (Составить такой график позволит демон `cron`; см. главу 8.)

Примечательна также команда `snmpwalk`. Начав с указанного идентификатора OID (или, по умолчанию, с начала базы MIB), она выполняет в цикле запрос `get-next`. В результате формируется полный список доступных идентификаторов OID и их значений. Команда `snmpwalk` особенно удобна, когда необходимо идентифицировать новые идентификаторы OID для осуществления мониторинга из какой-нибудь модной программы управления предприятием.

[6] Архив CPAN (Comprehensive Perl Archive Network — глобальная сеть архивов Perl-программ) содержит великолепную коллекцию полезных модулей Perl. Обращайтесь по адресу `www.cpan.org`.

Таблица 19.3. Команды пакета NET-SNMP

Команда	Назначение
snmpdelta	Контролирует изменения SNMP-переменных во времени
snmpdf	Контролирует изменение дискового пространства на удаленном компьютере по протоколу SNMP
snmpget	Запрашивает у агента значение SNMP-переменной
snmpgetnext	Запрашивает значение следующей переменной последовательности
snmpset	Передает агенту значение SNMP-переменной
snmptable	Запрашивает таблицу значений SNMP-переменных
snmptranslate	Осуществляет поиск идентификаторов OID и их описаний в базе MIB
snmptrap	Генерирует сообщение о прерывании
snmpwalk	Просматривает базу MIB, начиная с заданного идентификатора OID

Ниже в качестве примера приведена усеченная версия вывода команды `snmpwalk`, полученного на компьютере `tuva`. "secret813community" — это групповая строка, а `-v1` — это аргумент, указывающий, что должна использоваться простая аутентификация:

```
$ snmpwalk -c secret813community -v1 tuva
SNMPv2-MIB::sysDescr.0 = STRING: Linux tuva.atrust.com 2.6.9-11.ELsmp #1
SNMPv2-MIB::sysUpTime.0 = Timeticks: (1442) 0:00:14.42
SNMPv2-MIB::sysName.0 = STRING: tuva.atrust.com
IF-MIB::ifDescr.1 = STRING: lo
IF-MIB::ifDescr.2 = STRING: eth0
IF-MIB::ifDescr.3 = STRING: eth1
IF-MIB::ifType.1 = INTEGER: softwareLoopback(24)
IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifType.3 = INTEGER: ethernetCsmacd(6)
IF-MIB::ifPhysAddress.1 = STRING:
IF-MIB::ifPhysAddress.2 = STRING: 0:11:43:d9:1e:f5
IF-MIB::ifPhysAddress.3 = STRING: 0:11:43:d9:1e:f6
IF-MIB::ifInOctets.1 = Counter32: 2605613514
IF-MIB::ifInOctets.2 = Counter32: 1543105654
IF-MIB::ifInOctets.3 = Counter32: 46312345
IF-MIB::ifInUcastPkts.1 = Counter32: 389536156
IF-MIB::ifInUcastPkts.2 = Counter32: 892959265
IF-MIB::ifInUcastPkts.3 = Counter32: 7712325
...
```

В этом примере команда отобразила кое-какую общую информацию о системе и статистические данные о работе трех ее сетевых интерфейсов: `lo0`, `eth0` и `eth1`. В зависимости от поддерживаемых агентом баз MIB, вывод команды `snmpwalk` может содержать сотни строк.

Сбор и графическое представление данных SNMP

Касающиеся сети данные наиболее ценятся в визуальном и хронологическом контексте. Очень важно иметь какой-нибудь способ для отслеживания и графического представления метрических данных о производительности, но выбор ПО для выполнения этой задачи не столь критичен.

Раньше одним из наиболее популярных пакетов для сбора и графического представления SNMP-данных была программа MRTG (Multi-Router Traffic Grapher — многомаршрутный визуальный анализатор трафика), созданная Тобиасом Утикером (Tobias Oetiker) в Швейцарском федеральном технологическом институте в Цюрихе. MRTG написана в основном на языке Perl, запускается регулярно демоном `cron` и может собирать данные из любого SNMP-источника. Каждый запуск программы приводит к сохранению новых данных и созданию новых графиков.

Недавно Тоби разработал новую программу — RRDTool, которая представляет собой прикладной набор инструментов для сохранения и графического представления метрических показателей производительности. Все ведущие решения для мониторинга (из тех, что распространяются с открытым исходным кодом) основываются на RRDTool, в том числе и наша любимая программа Cacti.

Cacti (`www.cacti.net`) имеет несколько привлекательных функциональных возможностей. Во-первых, она включает не требующую административного вмешательства базу данных фиксированного размера, в которой хранятся только данные, необходимые для создания графиков. Например, программа Cacti умеет сохранять одну выборку для каждой минуты дня, каждого часа недели и каждой недели года. Такая схема укрупнения позволяет предоставлять пользователям важную информацию, не требуя хранения несущественных деталей и затрат времени на администрирование базы данных.

Во-вторых, программа Cacti способна записывать значения и отображать графики их изменения во времени для любой SNMP-переменной. Вместе с агентом NET-SNMP программа Cacti обеспечивает мониторинг практически всех системных и сетевых ресурсов.

На рис. 19.1 показаны примеры графиков, созданных программой Cacti. Эти графики отображают среднюю нагрузку на сервер за несколько недель и трафик, проходящий через сетевой интерфейс, за день.

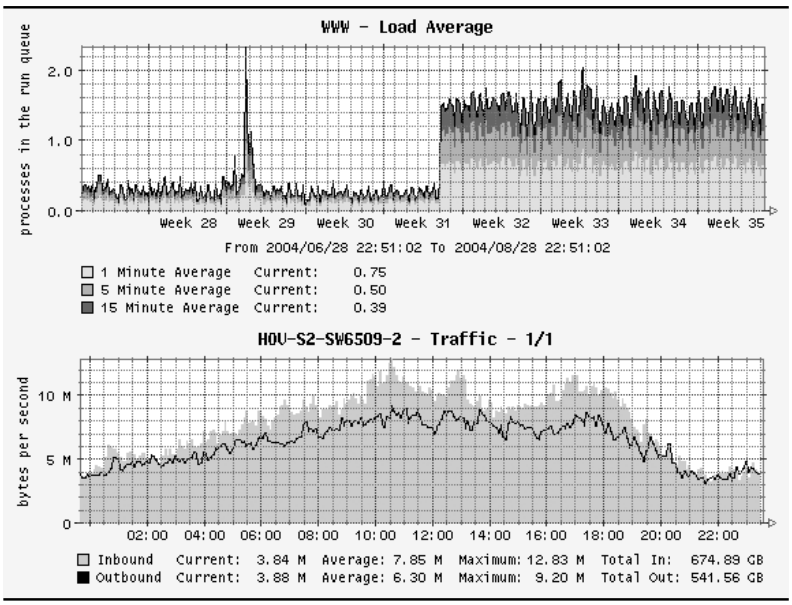


Рис. 19.1. Примеры графиков программы Cacti

Cacti пользуется популярностью благодаря простой Web-конфигурации и другим встроенным преимуществам RRDTool типа низкого объема обслуживания и красиво оформленных графиков. Ссылки на текущие версии RRDTool и Cacti, а также других средств мониторинга можно найти на домашней странице Тоби Утикера, которая находится по адресу `www.rrdtool.org`.

Nagios: основанный на событиях мониторинг SNMP и служб

Программа Nagios специализируется на составлении отчетов об ошибках в реальном времени. Она может собирать любые метрические показатели SNMP (и сотен других сетевых служб) и уведомлять о наметившихся сбойных ситуациях. Хотя помочь выяснить насколько возросла загруженность сети за последний месяц Nagios не может, зато она может уведомить о сбое Web-сервера.

В дистрибутив программы Nagios входят утилиты, позволяющие следить за различными "проблемными" компонентами сети. При желании можно создать свои собственные, дополнительные утилиты на языке Perl или даже C. Что касается методов уведомления, то Nagios может отправлять сообщения по электронной почте, генерировать Web-отчеты и отсылать уведомления на пейджер через модем. Как и в случае утилит мониторинга, легче разработать свой собственный метод.

Помимо отправки уведомлений о сбоях в работе служб программа Nagios также сохраняет все данные в специальном архиве. Она предоставляет несколько мощных интерфейсов для генерации отчетов, которые отслеживают показатели доступности и производительности. Во многих организациях Nagios используется для измерения совместимости с соглашениями об уровне обслуживания: на рис. 19.2, например, продемонстрированы показатели доступности сервера DNS.

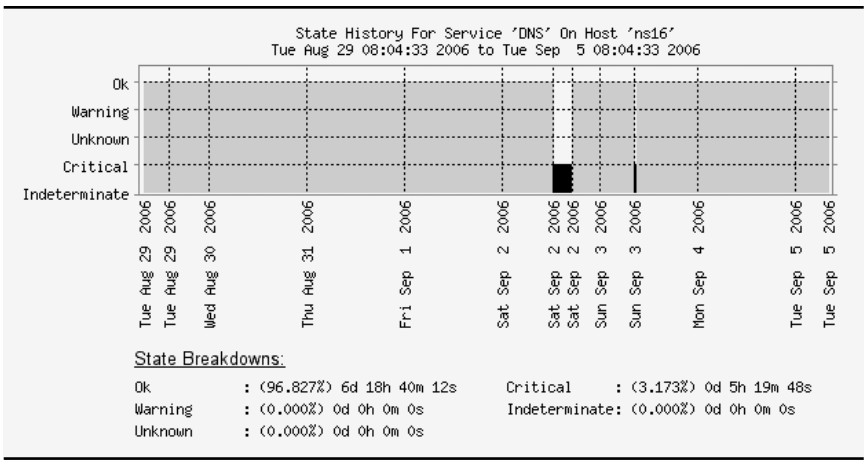


Рис. 19.2. Показатели доступности сервера, отображаемые программой Nagios

Программа Nagios очень хорошо работает в сетях, где число хостов и контролируемых устройств не превышает 1000. Она легко настраивается и расширяется, а также предоставляет такие мощные функциональные возможности, как избыточность, дис-

танционный мониторинг и распространение уведомлений. Тем, кто не может позволить себе приобрести коммерческое средство управления сетями, следует всерьез рассмотреть вариант использования программы Nagios. Дополнительную информацию можно найти по адресу `www.nagios.org`.

Коммерческие системы сетевого управления

Сотни компаний продают программное обеспечение для управления сетями, и каждую неделю на рынке появляются новые продукты. Мы не будем рекомендовать конкретные продукты (за время подготовки книги к печати все может измениться), а просто перечислим функциональные возможности, которые следует искать в системе управления сетью.

- **Возможность получения различных видов данных**. Средства управления должны уметь получать данные не только по протоколу SNMP. Многие программы позволяют принимать данные от большинства других сетевых служб, например выполнять SQL-запросы, обращаться к DNS и Web-серверам.
- **Качество пользовательского интерфейса**. Дорогостоящие системы обычно позволяют выбирать между имеющимся графическим интерфейсом или Web-интерфейсом. Программы, разработчики которых учитывают конъюнктуру рынка, поддерживают XML-шаблоны для форматирования данных. Важно, чтобы интерфейс позволял представлять информацию в наглядном и понятном виде.
- **Стоимость**. Некоторые программы имеют завышенную цену. Пакет OpenView компании Hewlett-Packard является одним из наиболее дорогостоящих, но в то же время одним из самых распространенных. Для многих компаний престижно, что их сети управляются высококачественной коммерческой системой. Если для вашей организации это не так важно, обратите внимание на бесплатное программное обеспечение, в частности программы Cacti и Nagios.
- **Автоматическое обнаружение хостов**. Многие системы обладают способностью "обнаруживать" сеть. Посылая широковещательные эхо-пакеты, выполняя SNMP-запросы, просматривая таблицы маршрутизации и обращаясь к службе DNS, они могут идентифицировать все компьютеры и устройства в сети. Как правило, данная функция реализована достаточно хорошо, но не всегда удается получить точные результаты в сложных сетях или сетях с брандмауэрами.
- **Средства отчета**. Многие программы способны посылать предупреждения по электронной почте, выдавать сообщения на пейджеры и автоматически генерировать уведомления для популярных систем обнаружения ошибок. Убедитесь в том, что выбранная платформа позволяет гибко настраивать систему отчетности. Кто знает, какие устройства появятся в ближайшие годы?
- **Возможности конфигурирования**. Некоторые решения включают не только возможности мониторинга и оповещения. Они позволяют управлять конфигурацией компьютеров и устройств. Например, система Cisco Works посредством специального интерфейса разрешает пользователю не только проверять состояние маршрутизатора по протоколу SNMP, но и менять его конфигурацию. Поскольку данные о конфигурации устройства очень полезны при глубоком анализе сетевых проблем, мы предполагаем, что в будущем располагать средствами изменения конфигурации сетевых компонентов будет большинство программ.

19.11. Рекомендуемая литература

- Cisco Online. *SNMP Overview*. 1996. `www.cisco.com/warp/public/535/3.html`
- Mauro, Douglas R. and Kevin J. Schmidt. *Essential SNMP. 2nd Edition*. Sebastopol, CA: O'Reilly Media, 2005.
- SimpleWeb. *SNMP and Internet Management Site*. `www.simpleweb.org`.
- Stallings, William. *Snmp, Snmpv2, Snmpv3, and Rmon 1 and 2 (3rd Edition)*. Reading, MA: Addison-Wesley, 1999.

Могут оказаться полезными и перечисленные ниже документы RFC. Вместо фактических названий этих документов мы приводим краткое описание их содержимого, так как их названия могут представлять собой не вполне понятный набор модных словечек и жаргонизмов SNMP.

- RFC1155 — характеристики пространства данных SNMP (типы данных и т.д.).
- RFC1156 — определения базы MIB-I (описание идентификаторов OID).
- RFC1157 — протокол SNMP (Simple Network Management Protocol — простой протокол управления сетью).
- RFC1213 — определения базы MIB-II (описание идентификаторов OID).
- RFC3414 — основанная на пользователях модель безопасности для SNMPv3.
- RFC3415 — основанная на представлениях модель контроля доступа для SNMPv3.
- RFC3512 — конфигурирование устройств с SNMP (наилучший общий обзор).
- RFC3584 — сосуществование различных SNMP-версий на практике.

19.12. Упражнения

19.1. Вы ищете источник проблемы в сети, и команда `netstat -rn` выдает показанные ниже результаты. В чем проблема и с помощью какой команды ее можно устранить?

```
Destination   Gateway  Genmask       Flags  MSS  Window  irtt  Iface
128.138.202.0 0.0.0.0  255.255.255.0 U      40   0       0     eth0
127.0.0.0     0.0.0.0  255.0.0.0     U      40   0       0     lo
```

☆ 19.2. Напишите сценарий, который будет контролировать заданную группу компьютеров и сообщать администратору по электронной почте, если какой-нибудь компьютер не реагирует на эхо-запросы в течение заданного времени. Не задавайте список компьютеров, почтовый адрес и величину интервала времени в тексте сценария.

☆ 19.3. Поэкспериментируйте с изменением сетевой маски одного из компьютеров локальной сети. Продолжит ли компьютер нормально работать? Все ли компьютеры доступны в сети? Можно ли с других компьютеров получить доступ к нему? Поддерживается ли широковещательный режим (т.е. запросы протокола ARP и пакеты DHCP)? Объясните полученные результаты. (Необходим доступ с правами суперпользователя.)

☆ 19.4. Воспользуйтесь командой `traceroute` для обнаружения маршрутов в своей сети.

а) Сколько переходов требуется сделать на пути к серверу?

б) Присутствуют ли маршрутизаторы между компьютерами, на которых у вас есть учетные записи?

в) Можно ли обнаружить узкие места в сети?

г) Есть ли в сети компьютеры с несколькими IP-адресами?

☆☆ 19.5. Составьте базу MIB, содержащую все переменные, которые вам как администратору Linux может понадобиться запрашивать или устанавливать. Предусмотрите возможность расширения этой базы данных.

☆☆ 19.6. Воспользуйтесь командой `tcpdump` для перехвата трафика перечисленных ниже протоколов. В случае протокола TCP укажите начальные и конечные пакеты. (Необходим доступ с правами суперпользователя.)

а) ARP;

б) эхо-запросы и эхо-ответы протокола ICMP;

в) SMTP;

г) FTP и FTP-DATA;

д) DNS (запрос доменного имени);

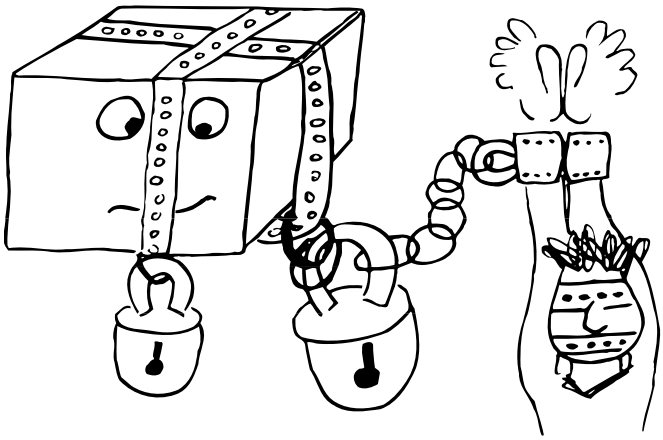
е) NFS.

☆☆ 19.7. Настройте программу Cacti на отображение графика пакетов, посылаемых локальному маршрутизатору и получаемых от него. Необходимо, чтобы агент SNMP опрашивал маршрутизатор; для этого следует узнать SNMP-имя маршрутизатора.

Book_LinuxAdmHand.indb 740 17.04.2007 13:56:07

Глава 20

Безопасность



Вышедший в 1983 г. кинофильм *War Games* (“Военные игры”) с Мэтью Бродериком (Matthew Broderick) в главной роли является прекрасной демонстрацией начала эры компьютерных “взломов”. В этом фильме герой Дэвид Лайтмен, применяя несколько хорошо известных злонамеренных технологий, таких как военный набор, социальная инженерия, проникновение в телефонные системы (т.е. “фрикинг”) и угадывание паролей, сначала начинает, а потом останавливает ядерную атаку. Фильм *War Games* привлек к вопросу компьютерной безопасности внимание общественности и сделал слово “хакер” синонимом выражения “тинэйджер, который разбирается в компьютерах и использует свои знания для того, чтобы напакостить”. Этот первый фильм был только началом того, что сегодня стало глобальной проблемой.

Пять лет спустя в мире компьютеров и сетевых технологий началась новая эра, когда Роберт Моррис-младший (Robert Morris, Jr.) спустил на человечество своего знаменитого “Internet-червя”. До того момента сеть Internet переживала “век невинности”. Вопросы безопасности, заботившие администраторов, звучали в основном так: “А что, если...?”. Серьезным происшествием считалась поимка пользователя, узнавшего пароль администратора и прочитавшего чужую почту (часто не по злому умыслу, а просто для самоутверждения).

“Червь” Морриса отнял у администраторов тысячи часов, но зато значительно повысил бдительность пользователей Internet в вопросах безопасности. Все мы просто еще раз вспомнили народную мудрость: чем выше заборы, тем добрее соседи. В результате появился целый ряд прекрасных инструментальных средств для системных администраторов (и была основана организация, предназначенная для борьбы с сетевым вредительством).

Согласно результатам опроса *Computer Crime and Security Survey* (Компьютерные преступления и безопасность)[1], проведенного в 2006 г. Институтом компьютерной безопасности совместно с ФБР, большинство организаций тратит на защиту своих систем менее 5% выделяемого на IT-нужды бюджета. А это значит, что на бреши в системе защиты компьютеров уходит громадная сумма в 54 494 290 долларов (174 103 долларов в каждой организации). Эти цифры свидетельствуют о серьезной обеспокоенности вопросами безопасности в организациях. Конечно, защита не всегда обязательно подразумевает трату денег: бдительный администратор способен выследить и остановить любого злоумышленника.

Безопасность нельзя купить в магазине или заказать в сервис-центре. Соответствующие коммерческие продукты и услуги могут быть частью предлагаемого решения для конкретной организации, но это не панацея.

Достижение приемлемого уровня безопасности требует огромного терпения, постоянной бдительности, немалых знаний и упорства. Это касается не только администраторов, но также всех пользователей и управляющего персонала. Системный администратор несет личную ответственность за безопасность системы и степень образованности пользователей в данном вопросе. Он должен знать современные технологии защиты, быть подписчиком соответствующих групп новостей и нанимать экспертов в области компьютерной безопасности, если его знаний оказывается недостаточно для решения возникшей проблемы.

20.1. Безопасна ли ОС Linux?

Увы! Операционную систему Linux нельзя назвать безопасной. Впрочем, как и любую другую систему, имеющую выход в сеть. Для создания абсолютно непробиваемой защиты необходимо изолировать[2] компьютер от всех устройств доступа. Некоторые считают, что компьютер также нужно поместить в специальную комнату, стены которой не пропускают электромагнитное излучение (попробуйте поискать в Google ключевые слова “клетка Фарадея”). Ну, кто может себе это позволить?

Кое-что для повышения надежности системы, разумеется, сделать можно. Но идеальная безопасность все же недостижима, ибо в модели безопасности Linux есть несколько фундаментальных изъянов, которые невозможно преодолеть.

- Как и UNIX, операционная система Linux ориентирована, прежде всего, на удобство применения, что отнюдь не предполагает естественность и простоту ее защиты. Концепция Linux заключается в обеспечении удобного манипулирования данными в сетевой многопользовательской среде.
- Стратегия защиты в Linux, если только она не реализована тщательным образом, по сути, предполагает всего два варианта статуса пользователя: пользователь без привилегий или пользователь root. Такие средства Linux, как, например, выполнение программ с установленным битом смены идентификатора пользователя, предназначены для обеспечения привилегированного доступа ко всем вычислительным ресурсам системы. Благодаря пакету “Security-Enhanced Linux”, разрабатываемому

[1] Этот опрос проводится ежегодно; дополнительную информацию можно найти по адресу `www.gosci.com`.

[2] Конечно, появление беспроводных сетей привело к возникновению новых проблем безопасности. В данном контексте “изолировать” означает “отключить сеть”.

Агентством национальной безопасности США (National Security Agency — NSA), были достигнуты определенные успехи в отношении реализации более точных средств контроля доступа. Но в большинстве случаев, незначительные огрехи в системе защиты все равно могут приводить к нарушению нормального функционирования целых систем.

- Linux разрабатывается большим сообществом программистов. Все они имеют разную квалификацию, по-разному относятся к своей работе и обладают неодинаковыми знаниями о строении операционной системы и ее особенностях. Поэтому даже выпущенные с самыми благими намерениями, самые современные средства защиты могут приводить к появлению новых "дыр". С другой стороны, исходный код Linux доступен каждому, и тысячи людей могут проверить каждую строчку этого кода на предмет наличия ошибок и делают это. Считается, что это приводит к существенному повышению безопасности по сравнению с закрытыми операционными системами, где доступ к "святая святых" — коду — имеет лишь небольшое число разработчиков.

Многие организации на одну-две версии отстают: либо по причине сложности локализации, либо потому, что они не заключили с поставщиком договор о сопровождении системы. В любом случае, даже когда "дыры" в системе защиты "залатаны", возможность проникновения хакера часто все равно за одну ночь не исчезает.

Раньше считалось, что по мере выявления и устранения брешей безопасность операционной системы Linux будет непрерывно повышаться. Суровая реальность оказалась иной. Сложность системного программного обеспечения стремительно растет, хакерская деятельность все больше приобретает черты организованной преступности, компьютеры оказываются все теснее связанными посредством сети Internet. Война переходит в новые измерения, и, похоже, победителей не будет. (Конечно, в мире Windows ситуация даже еще хуже).

Запомните такую формулу:

$$\text{Безопасность} = \frac{1}{1{,}072 \times \text{Удобство}}.$$

Чем безопаснее система, тем труднее пользователям работать в ней. Реализуйте предлагаемые в этой главе меры только после того, как тщательно изучите возможные трудности, которые могут в связи с этим возникнуть у пользователей.

20.2. Слабые места в системе защиты

В этой главе рассматриваются некоторые наиболее распространенные проблемы, связанные с безопасностью Linux-систем, и стандартные контрмеры, позволяющие избежать этих проблем. Но прежде чем перейти к деталям, давайте посмотрим, что приводит к возникновению таких проблем в реальном мире. Большинство проблем с безопасностью можно отнести к следующим категориям.

Социальная инженерия

Пользователи (и администраторы) системы часто оказываются самым слабым звеном в цепочке защиты. Даже сегодня, когда вопросам безопасности уделяется повышенное

внимание, ничего не подозревающие пользователи с хорошими намерениями очень легко становятся жертвами обмана и выдают секретную информацию.

Эта проблема проявляется во множестве форм. Злоумышленники звонят своим жертвам без предупреждения и выдают себя за законных запутавшихся пользователей, пытающихся получить помощь в доступе к системе. Администраторы непреднамеренно публикуют секретную информацию на общедоступных форумах при выявлении и устранении проблем. Физические проникновения случаются, когда якобы настоящий сотрудник службы технического осмотра меняет провода в будке телефонных переключателей.

Термином "фишинг" называют попытки выуживания у пользователей информации при помощи мошеннических электронных сообщений, чат-сообщений и даже используемых в мобильной связи SMS-сообщений. Защититься от фишинга особенно сложно, потому что фальшивые сообщения часто включают специфическую информацию жертв, что делает их похожими на настоящие.

Социальная инженерия продолжает оставаться мощной хакерской технологией и является одной из самых сложных в плане нейтрализации угроз. Установленная в организации политика безопасности должна предусматривать обучение новых сотрудников. Регулярные встречи на уровне организации являются эффективным способом для предоставления сотрудникам информации о том, что можно и что нельзя говорить по телефону, о физической безопасности, о фишинге и о выборе паролей.

Чтобы повысить степень сопротивляемости организации к атакам с применением социальной инженерии, возможно, будет полезно сымитировать несколько таких атак самостоятельно. Однако прежде чем делать это, следует получить разрешение от руководства. Подобные действия выглядят очень подозрительно, если выполняются без соответствующей санкции. Кроме того, они также являются разновидностью внутреннего шпионажа и потому могут вызвать негодование, если не будут выполняться открыто.

Уязвимые места в ПО

За много лет в программном обеспечении Linux (включая сторонние программы, как коммерческие, так и бесплатные) было выявлено несметное число ошибок, связанных с безопасностью. Используя незаметные программистские просчеты или контекстные зависимости, хакерам удавалось манипулировать системой по своему усмотрению.

Переполнение буфера — это одна из самых распространенных программных ошибок, причем такая, которая имеет очень серьезные последствия. Разработчики часто выделяют в памяти определенный заранее объем временного пространства, называемого *буфером*, для хранения определенного блока информации. Если их код не предусматривает сравнения размера данных с размером контейнера, в который те, по идее, должны помещаться, выделенные ячейки памяти оказываются под угрозой перезаписи. Хитроумные хакеры получают возможность ввести тщательно скомпонованные данные, которые приведут к аварийному сбою в работе программы или, что даже еще хуже, к выполнению какого-нибудь произвольного кода.

К счастью, внушительное количество атак типа "переполнение буфера" за последние годы повысило степень внимания программистов к этому вопросу. Хотя проблемы с переполнением буферов по-прежнему возникают, их, как правило, очень быстро обнаруживают и исправляют, особенно в приложениях с открытым исходным кодом. В состав более новых систем программирования, таких как Java и .NET, входят механизмы, которые автоматически проверяют размеры данных и предотвращают переполнение буфера.

Проблемы с переполнением буфера являются подкатегорией более широкого класса отражающихся на безопасности программных ошибок, называемых уязвимыми местами в системе верификации входных данных. Почти все программы принимают от пользователей какие-нибудь входные данные (например, аргументы командной строки или HTML-формы). Если код обрабатывает такие данные, не проверяя тщательно их формат и содержимое на соответствие, возможны проблемы. Рассмотрим следующий простой пример:

```
#!/usr/bin/perl
# Пример ошибки при верификации введенных пользователем данных
open (HTMLFILE, "/var/www/html/$ARGV[0]") or die "trying\n";
while (<HTMLFILE>) { print; }
close HTMLFILE
```

Цель этого кода, скорее всего, такова: отображать содержимое некоего HTML-файла в каталоге `/var/www/html`, который является стандартным корневым каталогом документов для Apache-сервера `httpd` на серверах Red Hat. Код принимает имя файла от пользователя и передает его в виде аргумента функции `open`. Однако если какой-нибудь злонамеренный пользователь введет в качестве этого аргумента `/../../etc/shadow`, он сможет увидеть содержимое файла `/etc/shadow`!

Что может сделать в этом случае администратор, чтобы предотвратить такую атаку? Немногое, по крайней мере, до тех пор, пока разработчик не выявит ошибку и не выпустит "заплату" для ее исправления. Знать обо всех последних "заплатах" и новостях в сфере безопасности — это святая обязанность любого администратора. В большинство дистрибутивов входят автоматизированные утилиты для создания "заплат", такие как утилита `yum` в дистрибутиве Fedora и утилита `apt-get` в дистрибутиве Debian и Ubuntu. Пользуйтесь этими утилитами для защиты своей системы от проблем типа уязвимых мест в программном обеспечении.

Конфигурационные ошибки

Многие компоненты программного обеспечения можно сконфигурировать в режиме полной или частичной безопасности. К сожалению, поскольку программное обеспечение разрабатывается так, чтобы им было удобно пользоваться, по умолчанию чаще всего используется второй вариант. Хакеры часто вламываются в системы, иезуитски эксплуатируя функциональные возможности, которые с миссионерской щедростью были предоставлены разработчиками в надежде сделать работу пользователей удобнее и гуманнее: учетные записи без паролей, глобальный совместный доступ к жестким дискам, незащищенные базы данных, и т.д., и т.п.

Типичным примером создания уязвимого места в конфигурации хоста является разрешение системам Linux загружаться, не запрашивая пароля начального загрузчика. Как LILO, так и CRUB во время установки можно сконфигурировать так, чтобы они запрашивали пароль, но администраторы практически всегда пропускают эту опцию. Пропуск этой опции делает систему открытой для физических атак. Однако это также может служить прекрасным примером необходимости в поиске баланса между безопасностью и удобством использования. Запрашивание пароля означает, что в случае случайной перезагрузки системы (например, в результате перебоя в электропитании), для ее запуска потребуется личное присутствие администратора.

Одним из наиболее важных шагов при обеспечении безопасности системы является просто убедиться в том, что, заботясь о благополучии пользователей, вы не пригласи-

ли на вечеринку монстра в маскарадном костюме. Проблемы этой категории легче всего отыскивать и устранять, хотя их может быть очень много и потому не всегда очевидно, что именно следует проверить. Средства сканирования портов и системы на наличие уязвимых мест, описываемые позже в этой главе, могут помочь заинтересованному администратору отыскивать и устранять проблемы до того, как они причинят вред системе.

20.3. Сертификаты и стандарты

Если предмет обсуждения этой главы кажется вам страшным, не волнуйтесь. Компьютерная безопасность — это сложная и обширная тема, о чем свидетельствует несметное количество посвященных ей книг, Web-сайтов и журналов. К счастью, для определения объема и организации доступной информации было сделано много. Сегодня существуют десятки стандартов и сертификатов; администраторам систем Linux остается только следовать содержащимся в них рекомендациям.

Один из самых основных философских принципов в информационной безопасности неофициально называется “триадой CIA”.

Акроним “CIA” расшифровывается так:

- C = Confidentiality = Конфиденциальность
- I = Integrity = Целостность
- A = Availability = Доступность

Под конфиденциальностью данных подразумевается соблюдение секретности данных. По сути, доступ к информации должен быть только у тех, кому разрешено иметь его. Аутентификация, контроль доступа и шифрование — все это компоненты конфиденциальности. Нарушением конфиденциальности считается, например, проникновение хакера в систему Linux и кража базы данных, содержащей контактную информацию клиентов.

Под целостностью подразумевается аутентичность информации. Технология целостности данных гарантирует, что информация является действительной, и не была изменена никаким несанкционированным образом. Она также проверяет степень надежности источников информации. Когда защищенный Web-сайт предоставляет подписанный SSL-сертификат, он тем самым доказывает пользователю не только, что отправляемая с него информация шифруется, но и также, что его подлинность была проверена авторитетным центром сертификации (таким как VeriSign или Equifax). Технологии типа PGP и Kerberos тоже гарантируют целостность данных.

Информация должна быть доступна санкционированным пользователя всегда, когда она им нужна, иначе в ее наличии нет никакого смысла. Простои, случающиеся не из-за злоумышленников, такие как те, что возникают из-за ошибок администраторов или перебоев в электропитании, тоже относятся к категории проблем, связанных с доступностью. К сожалению, доступность часто игнорируется до тех пор, пока что-нибудь пойдет не так.

Принципы CIA следует учитывать всегда, когда проектируешь, реализуешь или обслуживаешь системы Linux. Как говорится в одной старой пословице, “безопасность — это процесс”.

Сертификаты

Этот ускоренный курс по принципам CIA является всего лишь кратким введением в более обширную область информационной безопасности. В крупных корпорациях часто нанимают на полную ставку много сотрудников для выполнения работы, состоящей исключительно в охране информации. Чтобы завоевать доверие в этой области и совершенствовать свои знания, эти профессионалы посещают специальные курсы и получают специальные сертификаты.

Одним из наиболее известных сертификатов в области безопасности является CISSP (Certified Information Systems Security Professional — сертифицированный профессионал в области систем информационной безопасности). Его выдает Международный консорциум сертификации в области систем информационной безопасности, сокращенно называемый $(ISC)^2$. Одной из главных приманок CISSP является выпускаемый $(ISC)^2$ так называемый "общий источник знаний" (common body of knowledge — CBK), который, по сути, представляет собой справочник лучших технологий в сфере информационной безопасности. Он охватывает такие темы, как правовые аспекты информационной безопасности, криптография, аутентификация, физическая безопасность и многое другое, и является замечательным пособием для интересующихся вопросами информационной безопасности людей.

За что критикуют CISSP, так это за слишком большой размах и, как следствие, поверхностность получаемых знаний. Так много тем в CBK и так мало времени! Чтобы решить эту проблему, в $(ISC)^2$ разработали специализированные CISSP-программы, фокусирующиеся на архитектуре, проектировании и управлении. Эти специальные программы сертификации делают основной курс CISSP более углубленным.

GIAC (Global Information Assurance Certification — сертификат специалиста в области глобальной информационной безопасности) — это сертификат, который был разработан в 1999 г. Институтом системного администрирования, сетевых технологий и безопасности (The System Administration, Networking, and Security — SANS). Его получение предусматривает три десятка отдельных экзаменов, посвященных информационной безопасности, с поделенными на пять категорий тестами. Сложность курсов варьируется, начиная от курса средней сложности, подразумевающего прохождение двух экзаменов и позволяющего получить сертификат GISF, и заканчивая насыщенным курсом, позволяющего получить сертификат уровня эксперта GSF. Курс на получение сертификата GSF считается самым сложным в компьютерной индустрии. Многие из входящих в него экзаменов требуют наличия специфических технических знаний и определенного опыта.

И, наконец, CISA (Certified Information systems Auditor — дипломированный аудитор информационных систем) — это курс, позволяющий получить сертификат в области аудита и процессов. В этом курсе основное внимание уделяется бизнес-целостности, процедурам, мониторингу и другим связанным с управлением вопросам. Некоторые считают получение сертификата CISA промежуточным этапом, необходимым для успешного выполнения роли специалиста по безопасности в организации. Одним из наиболее привлекательных аспектов этого курса является то, что он подразумевает сдачу одного единственного экзамена.

Хотя получение сертификатов — это личное дело каждого, их применимость в сфере бизнеса неоспорима. Все больше и больше компаний сегодня воспринимают наличие сертификатов как показатель профессионализма. Во многих организациях сотрудникам с сертификатами предлагается более высокая заработная плата и более высокие должности. Если вы решите получить какой-нибудь сертификат, сообщите об этом в своей организации и попросите, чтобы вам помогли оплатить связанные с этим расходы.

Стандарты

Из-за постоянно увеличивающейся зависимости от систем данных, были созданы законы и нормативные документы для упорядочения процесса управления секретной и критически важной для бизнеса информацией. Все основные законодательные проекты, такие как HIPAA, FISMA и Sarbanes-Oxley Act (закон Сарбейнса-Оксли), включают разделы, посвященные компьютерной безопасности. Реализация перечисленных в них требований иногда обходится дорого, но зато они вынуждают уделять должное внимание некогда игнорировавшемуся аспекту технологии.

К сожалению, нормативные документы изобилуют юридическими терминами и потому могут быть трудны для понимания. Большинство из них не содержит никаких конкретных указаний касательно того, как достичь перечисленных в них требований. Поэтому, чтобы помочь администраторам с этим, разрабатываются стандарты. Эти стандарты не привязываются ни к каким конкретным нормативным документам, но следование им обычно гарантирует соответствие правовым нормам. Изучение требований сразу всех различных стандартов может показаться ужасно сложной задачей, но в любом случае является чрезвычайно полезным занятием.

Стандарт ISO/IEC 17799 является, пожалуй, самым популярным в мире. Он впервые вышел в 1995 г. как британский стандарт; занимает 34 страницы и поделен на 11 разделов, в которых рассматриваются самые различные вопросы, начиная от стратегий и физической защиты и заканчивая контролем доступа. В каждом разделе описывается определенная задача, конкретные требования к ее выполнению и наилучшие способы и средства для ее решения.

Требования не являются техническими и могут быть выполнены любой организацией наиболее соответствующим ее потребностям образом. Что касается недостатков, используемые в стандарте общие формулировки создают у читателя ощущение безграничной гибкости. Критики жалуются на то, что отсутствие конкретики оставляет организации уязвимыми к атакам.

Несмотря на это, данный стандарт является одним из наиболее ценных документов в области информационной безопасности. Он помогает устранить часто довольно ощутимую "пропасть" между администраторами и разработчиками и сфокусировать их усилия на минимизации риска организации.

PCI DSS (Payment Card Industry Data Security Standard — стандарт защиты данных в области платежных карт) — это совсем другой "зверь". Он появился из-за потребности в улучшении защиты в сфере обработки кредитных карт, резко возросшей после ряда драматических инцидентов. Например, в июне 2005 г. компания CardSystems Services International обнаружила "потерю" 40 миллионов номеров карточек.

Министерство внутренней безопасности США подсчитало, что только в 2004 г. на устранение последствий "краж личностей" (т.е. удостоверяющих личность данных) было потрачено 52,6 миллиарда долларов. Конечно, не все эти случаи были напрямую связаны с рассекречиванием данных кредитных карт, но большая бдительность со стороны производителей, несомненно, имела бы положительный эффект. ФБР даже удалось отыскать связь между кражами данных кредитных карт и финансированием террористических группировок. В частности, к профинансированным таким образом инцидентам были отнесены взрывы бомб в Бали и в метро в Мадриде.

Стандарт PCI DSS является результатом совместных усилий таких известных производителей кредитных карт, как Visa и Mastercard, хотя в настоящее время совершенствуется исключительно компанией Visa. В отличие от стандарта ISO 17799, он распро-

страняется бесплатно и доступен для загрузки. Он посвящен в основном защите систем данных владельцев карт и содержит 12 разделов, в каждом из которых перечисляются необходимые для защиты требования.

Поскольку стандарт PCI DSS фокусируется на системах обработки данных кредитных карт, для организаций, не имеющих дела с данными кредитных карт, он не подходит. Однако для тех, которые имеют, строгое соответствие этому стандарту является обязательным для избежания больших штрафов и возможно даже уголовной ответственности. Найти этот документ можно на Web-сайте Visa в разделе, посвященном защите коммерческих данных.

Во многих организациях создается специальное программное обеспечение для удовлетворения какой-нибудь уникальной потребности, но из-за спешки система защиты часто продумывается недостаточно тщательно. Подверженность атакам типа переполнения буфера и внедрения SQL-операторов или сценариев является одной из самых распространенных ошибок в ПО, разрабатываемом внутри организации. К счастью, правительство США выпустило процедурный документ, помогающий проверять безопасность таких приложений. Разработанный Министерством обороны "Контрольный перечень тестов для проверки безопасности приложений" (Application Security Checklist) представляет собой детальное руководство по тестированию и проверке приложений на наличие уязвимых мест в их системе защиты. Текущая версия этого документа доступна для загрузки по адресу

```
iase.disa.mil/stigs/checklist
```

В отличие от других описывавшихся в этом разделе стандартов, этот документ содержит множество технических деталей типа отдельных команд, которые должны выполняться для проведения того или иного конкретного теста. Он не является таким же удобным для чтения и формальным как другие перечислявшиеся стандарты, но зато он очень полезен для организаций, имеющих потребность в разработке своих собственных программ.

20.4. Безопасность: ключевые аспекты и философия

В этой главе рассматривается множество аспектов компьютерной безопасности. В идеале все их нужно учитывать. Но когда времени или терпения не хватает, приходится идти на компромисс. Ниже описаны шесть наиболее важных аспектов, а также приведены общие правила соблюдения "техники компьютерной безопасности".

Фильтрация пакетов

Если Linux-система подключается к сети, где есть выход в Internet, *необходимо*, чтобы между этой системой и внешним миром стоял брандмауэр либо маршрутизатор, фильтрующий пакеты. В качестве альтернативы можно включить фильтрацию пакетов в ядре с помощью утилиты `iptables` (которая будет описываться в разделе 20.13). Какой бы ни была реализация, фильтр должен пропускать через себя трафик только тех служб, которые администратор конкретно хочет, чтобы предоставлялись или использовались в системе Linux.

Ненужные службы

Дистрибутивы Linux очень сильно отличаются в плане того, какие сетевые службы в них активизируются при стандартной установке. Большинство из них включает несколько служб, которые вообще не нужны. Проверить, какие службы включены, и отключить те, в которых нет абсолютно никакой необходимости, — это обязанность администратора. Начать лучше всего с команд `netstat` и `fuser`. Понять, какие же действительно службы может увидеть злоумышленник, можно просканировав систему при помощи сканера портов с какого-нибудь удаленного компьютера. Утилита `nmap`, описываемая в разделе 20.10, является просто идеальным средством для выполнения этой задачи.

Программные "заплаты"

Все основные дистрибьюторы Linux регулярно (как правило, несколько раз в месяц) выпускают "заплаты" для устранения новых брешей в защите системы. *Внимательно* следите за появлением "заплат", имеющих отношение к вашей версии Linux (и любым установленным в ней программам), и максимально оперативно инсталлируйте их. Помните: раз появилась "заплата", значит, есть люди, которые уже несколько недель знают о существовании соответствующей дыры.

Резервные копии

Регулярно создавайте резервные копии всех систем, чтобы в случае инцидента можно было свести ущерб к минимуму. Никакие технологии зеркального дублирования, "горячей" замены или RAID не отменяют необходимость резервного копирования. Подробнее об этом рассказывалось в главе 9.

Пароли

Все мы любим простые правила. Вот одно из них: у *каждой* учетной записи должен быть пароль, который трудно угадать. Но даже самые лучшие пароли нельзя передавать через Internet в виде обычного текста. Поэтому если в системе допускается дистанционная регистрация, следует применять SSH или какой-нибудь другой безопасный механизм удаленного доступа (см. раздел 20.11).

Бдительность

Чтобы быть уверенным в безопасности системы, следите за ее состоянием, сетевыми соединениями, таблицей процессов. Делать это нужно регулярно (желательно каждый день). Выполняйте почаще самостоятельные проверки при помощи соответствующих средств, которые будут описываться чуть позже в этой главе. Проблема всегда начинается с малого, а затем нарастает, как снежный ком, так что чем раньше будет обнаружена аномалия, тем меньшим окажется ущерб.

Общие принципы защиты

Эффективная система безопасности должна базироваться на здравом смысле. Ниже перечислены некоторые основные правила, которым нужно следовать.

- Не оставляйте без присмотра файлы, которые могут представлять интерес для хакеров и не в меру любопытных сослуживцев. Коммерческие тайны, персональные досье, бухгалтерские ведомости, результаты выборов и т.д. — за всем этим нужен глаз да глаз. Гораздо надежнее зашифровать данные, чем просто пытаться предотвратить несанкционированный доступ к ним.
- В организации должен существовать порядок работы с секретной информацией. Некоторые рекомендации по этому поводу уже давались в этой главе, в разделе 20.3, и еще будут приводиться в главе 30.
- Старайтесь, чтобы в системе не было мест, где хакеры могли бы закрепиться. Хакеры часто вламываются в одну систему, а затем используют ее как базу для взлома других систем. Иногда они применяют сеть, чтобы скрыть свои следы, пока атакуют настоящую цель. Общедоступные службы с лазейками в системе защиты, каталоги, доступные для записи по протоколу FTP в анонимном режиме, групповые учетные записи, неиспользуемые системы — вот основные уязвимые места.
- Устанавливайте ловушки для обнаружения вторжений или попыток вторжения. Такие утилиты, как `samhain`, `xinetd` и John the Ripper (которые будут описываться в разделе 20.10), помогут предупреждать возможные проблемы.
- Внимательно следите за отчетами, которые генерируются этими утилитами. Незначительная проблема, проигнорированная в отчете, к моменту получения следующего отчета может перерасти в катастрофу.
- Учитесь защищать системы своими силами. Применяйте традиционные технологии, обучайте пользователей, руководствуйтесь, в конце концов, здравым смыслом. В случае необходимости приглашайте специалистов со стороны, но при этом тщательно контролируйте их работу.
- Постоянно следите за тем, не появились ли отклонения от нормального хода работы системы. Обращайте внимание на все необычное, например на непонятные журнальные сообщения или изменение характера использования какой-либо учетной записи (резкий рост активности, работа в необычное время, использование учетной записи во время отпуска ее владельца).

20.5. Проблемы защиты файлов `/etc/passwd` и `/etc/shadow`

Очень часто источником неприятностей является плохое управление паролями. В файлах `/etc/passwd` и `/etc/shadow` содержатся данные о том, кто может входить в систему, так что можно сказать, что эти файлы представляют собой “передовую линию защиты системы от захватчиков”. Их нужно вести с особой тщательностью, стараясь не допускать ошибок и не загромождать устаревшими данными.

📖 О файле `passwd` более подробно рассказывалось в разделе 6.1.

На унаследованных системах во втором поле файла `/etc/passwd` содержалась строка, которая представляла собой зашифрованный пароль пользователя. Для того чтобы могли работать такие команды, как `ls` и ей подобные, файл `/etc/passwd` должен был быть открыт для чтения. Из-за этого зашифрованная строка с паролем была доступна всем пользователям системы. Злоумышленнику ничего не стоило зашифровать целый

словарь и сравнить результаты со строками в файле /etc/passwd. При совпадении сравниваемых элементов злоумышленник получал пароль.

Насколько это опасно? В 80-е гг. прошлого столетия существовал по крайней мере один способ очень быстрой расшифровки паролей[3], но рядовому хакеру приходилось довольствоваться библиотечной функцией crypt[4] для шифрования слов из словаря с целью их последующего сравнения. В то время "быстродействующий" компьютер мог выполнять порядка нескольких сотен операций шифрования в секунду. Для сравнения, сегодня методом "грубой силы" хакеры могут взломать любой 56-битный ключ DES менее чем за 24 часа. К счастью, в современных системах UNIX и Linux функция crypt не используется, и защита является гораздо более надежной.

Такие данные пугают и указывают на необходимость ограничения доступа пользователей к зашифрованным строкам паролей. Самый распространенный способ — поместить пароли в отдельный файл, доступный только суперпользователю, а остальную часть файла /etc/passwd оставить без изменений. Файл, содержащий информацию о паролях, называется *файлом скрытых паролей* (/etc/shadow). В большинстве современных дистрибутивов Linux такой файл и скрытые пароли используются по умолчанию.

Проверка и выбор паролей

В Linux пользователи могут задавать собственные пароли. Это, конечно, очень удобно, но влечет за собой массу проблем, связанных с безопасностью. Выделяя пользователям учетные записи, обязательно давайте им указания о том, как правильно выбрать пароль. Следует предупредить пользователей о недопустимости выбора их фамилий или инициалов, имен детей и супругов, а также слов, которые можно найти в словаре. Пароли, сконструированные на основе таких личных данных, как номера телефонов и адреса, не менее легко поддаются расшифровке.

Рекомендуется выбирать пароль, состоящий не менее чем из восьми знаков, среди которых должны встречаться цифры, знаки препинания, а также прописные и строчные буквы. Бессмысленные сочетания знаков, слогов, первые буквы слов легко запоминаемой фразы — вот самые лучшие пароли. При этом легко запоминаемая фраза не должна быть одной из широко распространенных. Лучше придумать свою собственную. Советы, приводившиеся в разделе 3.3, посвященном выбору пароля для пользователя root, подходят и для паролей обычных пользователей.

Очень важно регулярно (желательно каждый день) проверять, у всех ли учетных записей имеется пароль. В записях файла /etc/shadow (или /etc/passwd, если в системе не используются скрытые пароли), содержащих описания псевдопользователей вроде daemon (такие пользователи являются владельцами некоторых системных файлов, но, естественно, никогда не регистрируются в системе), в поле пароля должна стоять звездочка (*) или восклицательный знак (!). Эти символы не соответствуют ни одному паролю и, таким образом, предотвращают использование учетной записи.

[3] В 1984 г. Эви Немет с помощью суперкомпьютера HEP нашла ключ к шифру Диффи-Хеллмана, которым часто пользовались при работе по алгоритму DES (Data Encryption Standard — стандарт шифрования данных). Считалось, что алгоритм DES математически надежен; тем не менее при использовании ключей малой длины достигается относительно невысокая степень секретности.

[4] Не следует путать ее с командой crypt, в которой используется другой, менее безопасный алгоритм шифрования.

Файл /etc/shadow в принципе не нуждается в обслуживании. Хотя существует один сценарий Perl, который позволяет проверять этот файл на наличие пустых (нулевых) паролей:

```
$ sudo perl -F: -ane 'print if not $F[1];' /etc/shadow
```

Этот сценарий выполняет проверку, отправляет результаты администратору и может запускаться с помощью демона cron. Чтобы упростить себе выполнение проверки правомерности внесенных в учетные записи изменений, можно написать сценарий, который будет ежедневно сверять файл /etc/passwd с его версией за предыдущий день (при помощи команды diff) и сообщать о выявленных различиях.

Устаревание паролей

В Linux можно реализовать механизм так называемого устаревания паролей, при котором пользователей заставляют периодически менять пароли. На первый взгляд, это хорошая идея, однако ее практическая реализация влечет за собой определенные проблемы. Не всякому пользователю по душе периодически менять пароль, поскольку это сопряжено с запоминанием нового пароля. Обычно для пароля выбирается простое слово, которое легко вводится и запоминается, а когда приходит время замены, многие пользователи, не желая себя утруждать, опять берут предыдущий пароль. Таким образом, дискредитируется сама идея. Модули PAM могут помочь принудить пользователей использовать сложные пароли и тем самым позволить избежать этой проблемы.

В любом случае, все пароли следует менять регулярно, особенно пароль пользователя root. При вводе пароль пользователя root должен легко "скатываться" с пальцев, чтобы его невозможно было определить, проследив за движением пальцев по клавиатуре. В нашей организации многие администраторы работают с утилитой sudo, не вводя пароль суперпользователя напрямую, но к его выбору мы все равно относимся с особой ответственностью.

📖 Об утилите sudo более подробно рассказывалось в разделе 3.4.

Управлять механизмом устаревания паролей позволяет программа chage. Используя эту программу, администраторы могут устанавливать минимальный и максимальный период между сменами пароля, даты истечения срока действия пароля, количество дней, за которое пользователи должны уведомляться об истечении срока действия их паролей, количество дней, на протяжении которого допускается бездействие прежде, чем учетные записи автоматически заблокируются, и др. Ниже показана команда, которая в качестве минимального периода между сменами пароля устанавливает 2 дня, в качестве максимального периода — 90 дней, в качестве даты истечения срока — 31 июля 2007 г., а в качестве периода, за который пользователь должен предупреждаться о приближении даты истечения срока действия — 14 дней:

```
$ sudo chage -m 2 -M 90 -E 2007-07-31 -W 14 ben
```

Групповые и совместно используемые учетные записи

Опасно, если учетная запись используется несколькими людьми. Групповые регистрационные имена (например, guest или demo) представляют собой удобную лазейку для хакеров. Во многих законодательных актах, таких как HIPAA, их даже запрещается использовать. Поэтому не разрешайте применять их в своей организации.

Нельзя допускать, чтобы пользователи делили учетные записи с членами семьи или друзьями. Если маленькому Джонни нужно войти в систему, чтобы решить домашнее задание, лучше предоставить ему для этой конкретной цели отдельное имя. В случае злоупотребления проще удалить регистрационное имя Джонни, чем папину учетную запись, особенно если речь идет о правительственном учреждении.

В некоторых организациях учетная запись root является групповой. Это слишком опасно! Рекомендуем контролировать предоставление прав суперпользователя с помощью утилиты `sudo` (см. раздел 3.4).

Пользовательские интерпретаторы команд

Не используйте сценарии как средство для неконтролируемой (беспарольной) регистрации в системе. На самом деле, при возникновении необходимости в беспарольной регистрации, возможно, лучше будет использовать вместо нее не требующие паролей SSH-ключи.

Привилегированные учетные записи

Единственная отличительная черта пользователя root состоит в том, что его идентификатор равен 0. Поскольку в файле `/etc/passwd` может быть несколько записей с таким идентификатором, то существует и несколько способов входа в систему в качестве суперпользователя.

Один из способов, который хакеры, получив доступ к интерпретатору команд суперпользователя, широко применяют для открытия "черного хода", — редактирование файла `/etc/passwd` посредством ввода в него новых учетных записей с идентификатором пользователя, равным 0. Поскольку такие команды, как `who` и `w`, работают с регистрационным именем, хранящимся в файле `/var/run/utmp`, а не с идентификатором владельца регистрационного интерпретатора, они не в состоянии разоблачить хакера, который выглядит как рядовой пользователь, хотя на самом деле зарегистрирован в системе в качестве суперпользователя.

Спасением от этого вероломства является мини-сценарий, подобный тому, который используется для поиска учетных записей, не имеющих пароля:

```
$ perl -F: -ane 'print if not $F[2];' /etc/passwd
```

Этот сценарий отображает любые записи файла `passwd`, в которых идентификатор пользователя не указан или равен 0. Сценарий можно адаптировать для поиска записей с подозрительными идентификаторами групп или идентификаторами пользователей, совпадающими с идентификаторами руководящих сотрудников организации.

Пристального внимания заслуживают также записи файла `passwd`, в которых нет имени пользователя либо вместо имени стоят знаки препинания. Эти записи могут показаться бессмысленными, но очень часто они позволяют свободно входить в систему.

Модули PAM

Подключаемые модули аутентификации (Pluggable Authentication Modules, PAM) были разработаны компанией Sun в качестве гибкого средства аутентификации пользователей. Долгие годы под аутентификацией в среде UNIX подразумевалось сопоставление регистрационных данных пользователей с их записями в файле `/etc/passwd`. Необходимость в усилении защиты и поддержке большего разнообразия средств аутентификации (например, интеллектуальных карточек) потребовала более гибкого подхода.

Сегодня определенные LDAP-модули PAM выполняют аутентификацию централизованно, используя глобальные аутентификационные каталоги.

Модуль Linux-PAM входит в состав всех основных дистрибутивов Linux и не имеет отношения к текущей реализации стандарта PAM компании Sun. Концепция проста: программам, выполняющим аутентификацию, достаточно "знать" лишь о том, что имеется некий модуль, который проведет аутентификацию за них. Система конфигурируется таким образом, чтобы модули можно было добавлять, удалять и перенастраивать в любое время. Вовсе не обязательно, чтобы тот или иной модуль был скомпонован (или даже существовал) на этапе компиляции утилиты. Благодаря этому модули PAM стали удобным средством системного администрирования.

Сегодня доступны десятки модулей PAM. Загрузить специальные модули и документацию к ним можно по адресу `www.kernel.org/pub/linux/libs/pam`.

Модули PAM настраиваются посредством файлов каталога `/etc/pam.d`. Для каждой службы в этом каталоге содержится отдельный файл с записями следующего вида:

```
тип_модуля управляющий_флаг путь_к_модулю аргументы
```

Поле *тип_модуля* может иметь значения `auth`, `account`, `session` или `password`. Модуль типа `auth` идентифицирует пользователя и может предоставлять ему право группового доступа. Модуль типа `account` выполняет действия, не связанные с аутентификацией, например предоставляет доступ в зависимости от времени суток. Действия, которые необходимо выполнить до или после того, как пользователь получит доступ к требуемой службе, реализуются модулем типа `session`. Наконец, модуль типа `password` используется, когда пользователь должен ввести аутентификационную информацию (например, пароль).

Поле *управляющий_флаг* имеет четыре возможных значения: `required`, `requisite`, `sufficient` и `optional`. Чаще всего используются флаги `required` и `optional`. Первый из них указывает на то, что для успешного продолжения работы модуль должен добиться положительного результата. Второй флаг говорит о том, что результат работы модуля несущественен.

В третьем и четвертом полях указываются путь к динамически загружаемому файлу модуля и аргументы его вызова. Если первым символом третьего поля является косая черта (/), то путь считается абсолютным. В противном случае содержимое поля добавляется к стандартному имени каталога: `/lib/security`.

PAM является одним из решений для связанных со сложностью паролей трудностей, которые описывались выше. Модуль `pam_cracklib` может принудительно делать так, чтобы пароли отвечали минимальным требованиям. Специфические детали очень сильно варьируются, поэтому используйте `grep`, чтобы отыскать нужный конфигурационный файл. Например, для гарантии того, что пароли пользователей не смогут быть получены с помощью программы John the Ripper, файл `/etc/pam.d/system-auth` на Fedora должен содержать такую строку:

```
password required pam_cracklib.so retry=3 minlen=12 difok=4
```

📖 О программе John the Ripper более подробно будет рассказываться в разделе 20.10.

При наличии такой строки PAM сверяет предлагаемые пользователями новые пароли со словарем и правилами `pam_cracklib` (для этого необходимо, чтобы присутствовала системная библиотека `libcrack` и системный словарь `/usr/lib/cracklib_dict`). В случае ввода пользователем пароля, не отвечающего требованиям `cracklib`, на экран выводиться сообщение об ошибке наподобие "Пароль является слишком простым".

Правила касательно аргументов для `cracklib` довольно сложны, но вот что означают аргументы в показанном выше примере.

- Аргумент `retry=3` указывает на то, что пользователю должны предоставляться три попытки для ввода правильного пароля.
- Аргумент `minlen=12` указывает минимальную длину пароля. Буквы в верхнем регистре, числа и знаки пунктуации интерпретируются библиотекой специальным образом и сокращают минимальную длину. При значении `minlen=12` самый короткий пароль, который может ввести пользователь, — это 8-, а не 12-значный пароль, но в таком случае он должен включить в него символы всех четырех доступных типов.
- Аргумент `difok=4` указывает, что, по крайней мере, четыре символа в новом пароле не должны совпадать с теми, чтобы были в старом пароле.

В современных дистрибутивах Linux модуль `pam_cracklib` присутствует и используется по умолчанию, а вот правила, касающиеся сложности паролей, обычно не включены.

20.6. Возможности POSIX

В системах Linux привилегии учетной записи пользователя root распределяются в соответствии с возможностями POSIX. Например, одна возможность позволяет процессу выполнять привилегированные операции с сетевыми сокетами и сетевыми интерфейсами, в то время как другая позволяет процессу устанавливать аппаратные опции. Эти возможности наследуются, маскируются и выдаются систематизированным образом, благодаря чему программы могут выполнять привилегированные задачи, не получая доступа ко всем привилегиям учетной записи root. Поскольку привилегии могут распределяться более точно, чем это было в традиционной системе UNIX, вероятность нарушения защиты и получения доступа уровня пользователя root является более низкой (по крайней мере, в теории).

На практике система распределения возможностей не представляет особого интереса для системных администраторов. Распознающее ее программное обеспечение не требует никакого специального внимания со стороны администратора: оно просто устанавливает себя так, чтобы функционировать в более ограниченном режиме. Но многие программы UNIX и Linux по-прежнему предполагают наличие традиционной модели всемогущего суперпользователя, отключить которую без определенных трудностей и жертв не возможно. Некоторые из настроек, выполняемых ядром Linux для обеспечения совместимости с традиционными программами UNIX, делают систему возможностей более уязвимой, чем ей следует быть на самом деле[5].

Вывод таков: возможности POSIX представляют собой хорошую тему для разговора за коктейлем, но имеют очень ограниченное применение в реальном мире. По большей части они могут игнорироваться, но есть одно важное исключение: ядро применяет “глобальный ограничивающий набор возможностей” (доступный и задаваемый в файле `/proc/sys/kernel/cap-bound`) каждый раз, когда выполняется какая-нибудь новая программа. Поэтому, имея желание отключить одну или более возможностей во

[5] Один технический специалист прокомментировал это так: “В действительности, система наследования возможностей никогда не работала так, как нужно, в Linux: она постоянно создает какие-нибудь проблемы”.

всей системе, это можно сделать, добавив их в этот глобальный ограничивающий набор. Подробнее об этом можно узнать на man-странице capabilities.

20.7. Программы с установленным битом смены идентификатора пользователя

Программы, выполняемые с установленным битом setuid (Set User ID — смена идентификатора пользователя), являются источником проблем для безопасности системы, особенно если речь идет о получении привилегий суперпользователя. Такие программы, поставляемые вместе с Linux, являются безопасными, по крайней мере, теоретически. Однако огрехи в защите обнаруживались в прошлом и, несомненно, будут обнаруживаться в будущем.

Самый надежный способ уменьшения количества проблем, вызванных сменой идентификатора, — сведение к минимуму числа программ с установленным битом setuid. Подумайте дважды, прежде чем инсталлировать такую программу, и вообще избегайте устанавливать этот бит для своих программ. В среднем в дистрибутивах Linux содержится около 35 программ с установленным битом смены идентификатора пользователя, хотя это значение варьируется от дистрибутива к дистрибутиву.

Нет правила, требующего, чтобы программы с установленным битом setuid запускались от имени суперпользователя. Если нужно всего лишь ограничить доступ к файлу или базе данных, достаточно добавить в файл /etc/passwd псевдопользователя, единственное назначение которого будет заключаться во владении нужными ресурсами. Следуйте обычным соглашениям о добавлении псевдопользователей: используйте низкое значение UID, поставьте в поле пароля звездочку и сделайте начальным каталогом псевдопользователя каталог /dev/null.

Выполнение программ с установленными битами setuid и setgid (Set Group ID — смена идентификатора группы) можно отключать в отдельных файловых системах с помощью опции -o nosuid команды mount. Хорошими кандидатами на это являются файловые системы, содержащие начальные каталоги пользователей или смонтированные из ненадежных административных доменов.

Полезно периодически сканировать диски на предмет выявления новых программ с установленным битом setuid. Хакер, взломавший систему, вполне может создать собственный командный setuid-интерпретатор, который облегчит ему последующий вход в систему. Ряд программ, описанных в разделе 20.10, позволяет обнаруживать такие файлы, но вполне подойдет и простая команда find, например:

```
/usr/bin/find / -user root -perm -4000 -print |
   /bin/mail -s "Setuid root files" netadmin
```

В данном случае пользователю netadmin по электронной почте посылается список всех файлов, принадлежащих пользователю root и имеющих установленный бит setuid.

20.8. Специальные права доступа

Многие файлы в Linux должны иметь специальные права доступа, чтобы не возникали проблемы, связанные с безопасностью. Некоторые поставщики выпускают дистрибутивы, в которых права доступа заданы в расчете на собственную “дружественную” среду разработки. Для пользователей такие установки могут оказать “медвежью услугу”.

Файлы /etc/passwd и /etc/group должны быть доступны для записи только владельцу (пользователю root) и его группе. Режимом доступа должен быть 644. А для файла /etc/shadow режим доступа должен выглядеть как 600 (т.е. этот файл не должен быть вообще доступен никаким рядовым пользователям):

```
-rw-------  1 root root  1835 May 8 08:07  /etc/shadow
```

Все эти файлы должны принадлежать какой-нибудь системной группе (обычно это root). Чтобы пользователи, не имея права записи в файлы /etc/passwd и /etc/shadow, могли изменять свои пароли, команда passwd (владельцем которой является пользователь root) выполняется с установленным битом setuid.

Исторически FTP-программы изобиловали брешами в системе защиты и небезопасными стандартными конфигурациями. Кроме того, FTP также передает данные удостоверения в виде открытого текста, что является давней проблемой этого протокола и делает его использование недопустимым в современном мире Internet.

О настройке FTP-сервера более подробно будет рассказываться в разделе 21.6.

Сегодня FTP заменили новые технологии, такие как SFTP, и существует лишь несколько случаев, в которых можно разрешать использование FTP. В этих редких случаях, однако следует очень внимательно изучать документацию программного обеспечения, дабы быть уверенным в том, что была выбрана правильная конфигурация. Если вы вынуждены разрешить анонимный FTP-доступ, тогда либо отключите каталоги, доступные для записи всем пользователям, либо регулярно проверяйте их на наличие несанкционированных и секретных файлов.

Файлы устройств для разделов жесткого диска являются еще одним потенциальным источником проблем. Наличие права чтения или записи такого файла равнозначно наличию аналогичных прав для любого элемента файловой системы этого раздела. Право чтения/записи должен иметь только суперпользователь. Члены группы иногда могут получать право чтения с целью выполнения резервного копирования. Прав доступа для категории “прочие” быть не должно.

20.9. Различные аспекты безопасности

В следующих разделах рассматривается ряд дополнительных вопросов, связанных с безопасностью. В основном речь пойдет либо о средствах, которые могут быть полезны администратору, либо об “антисредствах”, которые при отсутствии должного контроля могут стать оружием в руках хакеров.

Дистанционная регистрация сообщений

Система Syslog позволяет записывать журнальную информацию о процессах ядра и пользовательских процессах в файл, рассылать эту информацию группе пользователей или передавать на другой компьютер сети. Рассмотрите вариант выделения защищенного компьютера, который будет служить центральной регистрационной станцией, выполнять синтаксических анализ журнальных файлов и рассылать по электронной почте сообщений об интересующих событиях. Это не позволит хакерам заметать следы путем перезаписи и удаления журнальных файлов.

Подробнее о системе Syslog рассказывалось в главе 10.

Защищенные терминалы

Linux можно настроить так, чтобы пользователю `root` разрешалось регистрироваться в системе только с определенных, "защищенных" терминалов. Рекомендуется запрещать привилегированный вход в систему по таким каналам, как SSH, ВЧС и другие удаленные соединения.

Защищенные каналы обычно задаются в виде списка TTY-устройств в файле `/etc/securetty`. Можно также назначить непривилегированным пользователям разрешенные для них терминалы (файл `/etc/security/access.conf`) и разрешенное время работы в системе (файл `/etc/security/time.conf`).

Администраторы все равно смогут входить в систему обычным образом и с помощью `sudo` получать доступ к правам суперпользователя.

Файлы `/etc/hosts.equiv` и `~/.rhosts`

В файлах `hosts.equiv` и `~/.rhosts` компьютеры определяются как административно "эквивалентные" друг другу, что позволяет пользователям входить в систему (с помощью утилиты `rlogin`) и копировать файлы (посредством утилиты `rcp`) с одного компьютера на другой, не вводя пароль. Раньше, в "век невинности", подобная конфигурация была широко распространена, но со временем стало ясно, какую потенциальную угрозу она несет в себе.

К счастью, появление протокола SSH положило конец использованию таких небезопасных утилит как `telnet`, `rsh` и `rlogin`. Об этом протоколе более подробно будет рассказываться чуть позже в этой главе.

Некоторые программы, используемые взамен `rlogin` (включая SSH!), все же обращаются к файлам `.rhosts` и `/etc/hosts.equiv`, если их неправильно сконфигурировать. С целью повышения безопасности лучше всего для каждого пользователя (в том числе пользователя `root`) создать файлы `/etc/hosts.equiv` и `~/.rhosts` нулевой длины, причем без права записи. Гораздо легче определить, каково состояние файла в 3 часа ночи, существует ли он и не подвергался ли изменениям, чем оценить состояние несуществующего файла. Это поможет при выслеживании нарушителей и их попыток "взломать" систему.

Безопасность и NIS

Эти слова можно поставить рядом разве что в заголовке. NIS (Network Information Service — сетевая информационная служба) представляет собой разработанную компанией Sun систему распространения баз данных, которой многие организации пользуются для ведения и рассылки таких файлов, как `/etc/group`, `/etc/passwd` и `/etc/hosts`. К сожалению, ошибочна уже сама лежащая в основе этой системы идея "легкого доступа к информации", поэтому NIS является лакомым кусочком для хакеров.

📖 Подробнее о NIS рассказывалось в главе 17.

Более безопасный и надежный способ рассылки указанных файлов — создать учетную запись наподобие `netadmin` и помещать последние копии файлов в каталог `~netadmin`. После этого на каждом клиентском компьютере необходимо с помощью демона `cron` периодически запускать сценарий, предназначенный для копирования (с помощью утилиты `scp`), проверки и инсталляции файлов. Утилита `scp` является частью пакета SSH, о котором пойдет речь в разделе 20.11.

Безопасность и NFS

Протокол NFSv4 — это IETF-расширение более раннего протокола Sun. Он имеет более надежную защиту и ряд других преимуществ по сравнению с более ранними реализациями. Хотя этот протокол еще не закончен, его разработка идет полным ходом. Linux-ядра серии 2.6 уже включают его.

В более старых версиях NFS используется слабая модель безопасности. Дополнительные сведения о безопасности NFS приводились в конце раздела 16.1. С помощью команды `showmount -e` можно узнать, какие файловые системы экспортируются и кому именно. Для каждой экспортируемой файловой системы необходимо задать список управления доступом, все имена компьютеров в котором должны быть полностью определенными.

Безопасность и `sendmail`

`sendmail` — это огромная программа, которая запускается и выполняется от лица пользователя root, по крайней мере, первоначально. Вследствие этого она часто становилась мишенью для хакеров. Поэтому следует пользоваться только самой новой версией программы. К выпуску очередных версий чаще всего приводят как раз проблемы безопасности, так что ошибки, очевидно, есть во всех версиях `sendmail`, кроме самой последней.

Более подробные сведения о безопасности программы `sendmail` приводились в разделе 18.12. Информацию о конкретных выпусках можно найти на Web-сайте `www.sendmail.org`.

📖 Более подробно о программе `sendmail` рассказывалось в главе 18.

Безопасность и резервное копирование

Регулярное создание резервных копий является неотъемлемым компонентом любой стратегии защиты. Необходимо, чтобы все разделы диска регулярно архивировались на ленту, а некоторые резервные копии хранились вне стен организации. В таком случае, если произойдет серьезный инцидент, можно будет быстро восстановить работоспособность системы.

📖 О резервном копировании более подробно рассказывалось в главе 9.

Резервные копии тоже могут представлять угрозу безопасности. Украденная коллекция лент может развалить всю остальную систему защиты. Во избежание кражи резервные ленты лучше хранить в огнеупорном сейфе. Стоит рассмотреть вариант использования шифрования. При намерении подписать контракт на хранение резервных лент с какой-нибудь сторонней компанией, не помешает сначала лично осмотреть потенциальное место хранения.

Вирусы и черви

Linux считается наиболее устойчивой к вирусам системой. Существует лишь горстка вирусов (большая часть из которых является академическими по своей природе), но ни один из них никогда не причинял такого серьезного ущерба, какой так часто наносят вирусы в мире Windows. Тем не менее, это не мешает некоторым производителям антивирусных средств заявлять о поражении платформы Linux вредоносным ПО (конечно,

если не будут приобретены их антивирусные продукты по специальной символической цене).

Точная причина отсутствия зловредного ПО не ясна. Некоторые утверждают, что Linux просто имеет меньшую долю на рынке по сравнению со своими настольными конкурентами и потому не интересна авторам вирусов. Другие настаивают на том, что Linux защищает от самораспространяющихся червей и вирусов ее управляющая доступом среда.

В последнем утверждении есть доля правды. Поскольку Linux ограничивает доступ для записи к системным исполняемым файлам на уровне файловой системы, учетные записи непривилегированных пользователей не могут заразить остальную часть среды. Если только код вируса не выполняется от лица пользователя root, масштаб заражения является очень ограниченным. Главным принципом тогда должно быть: не использовать учетную запись пользователя root для выполнения повседневных операций.

Немного парадоксально, но существует одна причина, по которой может быть все-таки полезно установить антивирусное ПО на серверах Linux, — это защитить установленные в организации системы Windows от ориентированных специально на Windows вирусов. Почтовый сервер может проверять входящие электронные сообщения на наличие вирусов, а файловый сервер может проверять совместно используемые файлы на наличие инфекции. Однако такое решение должно дополнять настольную антивирусную защиту, а не заменять ее.

Программа ClamAV, написанная Томасом Коймом (Tomasz Kojm), является популярным бесплатным антивирусным продуктом для Linux. Она представляет собой широко применяемый GPL-набор антивирусных средств и включает сигнатуры для тысяч вирусов. Загрузить последнюю версию этой программы можно по адресу `www.clamav.net`.

Троянские кони

Троянские кони — это программы, назначение которых полностью противоречит тому, чего от них ожидают. В качестве примера можно привести программу `turkey`, которая когда-то распространялась в Usenet. Она обещала нарисовать на экране терминала индейку, а на самом деле удаляла файлы из начальных каталогов доверчивых пользователей.

Другим примером троянского коня является вызвавшая массу споров программа для защиты от копирования, поставлявшаяся на многих аудио CD-дисках Sony в 2004 и 2005 гг. Эта программа устанавливалась на системах Windows без согласия и ведома пользователя и приводила к появлению уязвимых мест, которые могли использоваться червями и вирусами.

Троянские кони периодически появляются в основных пакетах программ Linux. `sendmail`, `tcpdump`, OpenSSH и InterBase выпустили рекомендации касательно зловредного ПО в их продуктах. Эти троянские кони обычно содержать вредоносный код, который позволяет хакерам получать доступ к системам жертвы в любое удобное для них время. К счастью, большинство производителей исправляет погрешности в своих продуктах, и через неделю или две выпускает соответствующие рекомендации. Обязательно подписывайтесь и проверяйте списки рассылки на наличие новостей о пакетах сетевых программ, которые установлены на ваших компьютерах Linux.

Учитывая количество нападений, которые сообщество Linux пережило за последние годы, остается удивляться малому числу инцидентов с троянскими конями. Таким положением вещей мы обязаны скорости Internet-связи. Наиболее очевидные проблемы,

связанные с безопасностью, быстро выявляются и широко обсуждаются. Злонамеренные программы не задерживаются на популярных серверах сети Internet.

Можете быть уверены: любая программа, у авторов которой были нечистые намерения, будет выявлена и проанализирована в Internet. Если перед инсталляцией какой-то программы нужно узнать ее "репутацию", просто введите имя программы в любимой поисковой системе.

Руткиты

Самые хитроумные хакеры стараются скрывать свои следы и избегать обнаружения. Обычно они надеются продолжать пользоваться системой жертвы для нелегального распространения ПО, зондирования других сетей или проведения атак в других системах. Для того чтобы оставаться необнаруженными они часто используют средства, называемые "руткитами" (rootkit). Например, программа Sony скрывала себя от глаз пользователя при помощи именно такой технологии.

Руткиты — это программы и заплаты, которые скрывают важную системную информацию наподобие процесса, диска или сетевой операции. Они имеют множество разновидностей и варьируются по сложности от простых замененных приложений (таких как хакерские версии утилит `ls` и `ps`) до модулей ядра, которые практически невозможно обнаружить.

Централизованные программы обнаружения попыток проникновения, такие как `samhain` (которая описывается ниже), являются эффективным средством для мониторинга, т.е. проверки систем на наличие руткитов. Хотя доступны программы, позволяющие администраторам удалять руткиты, время, требующееся на выполнение тщательной чистки, лучше потратить на сохранение данных, форматирование диска и установку всего необходимого с нуля. Самые усовершенствованные руткиты знакомы с основными используемыми для их удаления программами и стараются обходить их.

20.10. Инструментальные средства защиты

Некоторые из занимающих много времени и упоминавшихся в предыдущих разделах задач можно автоматизировать при помощи распространяемых бесплатно утилит. Ниже будут рассмотрены наиболее интересные из них.

Nmap: сканирование сетевых портов

Утилита Nmap представляет собой сканер сетевых портов. Ее основное назначение — проверка указанных компьютеров на предмет того, какие TCP- и UDP-порты на них прослушиваются серверными программами[6]. За большинством сетевых служб закреплены "известные" порты, поэтому такая информация позволяет узнать, какие программы выполняются на компьютере.

Запуск Nmap — отличный способ узнать, как выглядит система в глазах того, кто пытается ее взломать. В качестве примера приведем отчет, выданный этой утилитой на компьютере Red Hap Enterprise Linux со стандартной конфигурацией:

[6] Как объяснялось в главе 12, порт — это нумерованный канал взаимодействия. IP-адрес идентифицирует весь компьютер, а комбинация IP-адреса и номера порта определяет конкретный сервер или сетевое соединение.

```
$ nmap -sT rhel.booklab.example.com
Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-15 23:48 EDT
Interesting ports on rhel.booklab.example.com (192.168.1.31):
(The 1668 ports scanned but not shown below are in state: closed)
PORT       STATE    SERVICE
22/tcp     open     ssh
25/tcp     open     smtp
111/tcp    open     rpcbind
631/tcp    open     ipp

Nmap run completed -- 1 IP address (1 host up) scanned in .158 second
```

Аргумент `-sT` по умолчанию сообщает `nmap` о необходимости подключиться к каждому TCP-порту на указанном хосте[7]. Как только соединение устанавливается, `nmap` немедленно отключается, что не очень корректно, зато безболезненно для правильно написанного сетевого сервера.

В показанном выше примере мы видим, что на хосте `rhel` выполняется несколько служб, которые, скорее всего, не используются и являются традиционным источником проблем с безопасностью: это `portmap` (`rpcbind`), CUPS (`ipp`) и, пожалуй, `sendmail` (`smtp`). Таким образом, потенциальные направления удара уже ясны.

Колонка `State` (состояние) в выводе `nmap` содержит запись `open` (открыт) для портов, с которыми связаны серверы, запись `unfiltered` (не фильтруется) для портов, состояние которых неизвестно, запись `closed` (закрыт) для портов без серверов и запись `filtered` (фильтруется) для портов, опросить которые не удалось из-за наличия промежуточного фильтра пакетов. Нефильтруемые порты являются обычным делом и, как правило, отображаются, только если `nmap` выполняет сканирование ACK. Вот, например, результат запроса к более защищенному серверу `secure.example.com`:

```
$ nmap -sT secure.example.com
Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-15 23:55 EDT
Interesting ports on secure.example.com (192.168.1.33):
(The 1670 ports scanned but not shown below are in state: closed)
PORT       STATE    SERVICE
22/tcp     open     ssh
1241/tcp   open     nessus

Nmap finished: 1 IP address (1 host up) scanned in 0.143 seconds
```

В данном случае очевидно, что компьютер сконфигурирован так, что на нем может использоваться SSH и сетевой сканер Nessus. Брандмауэр блокирует доступ к другим портам.

Помимо стандартного опроса TCP- и UDP-портов команда `nmap` способна проверять порты “по-хакерски”, не устанавливая с ними реального соединения. В большинстве случаев посылаются пакеты, которые выглядят взятыми из уже имеющегося TCP-соединения (это не квитирующие пакеты), а затем проверяются полученные в ответ диагностические пакеты. Таким способом можно обойти брандмауэр или программу мониторинга сети, которая контролирует появление сканеров портов. Если в вашей организации имеется брандмауэр (см. раздел 20.12), не поленитесь проверить его в разных режимах сканирования.

[7] На самом деле по умолчанию проверяются только привилегированные (т.е. те, у которых номера меньше 1024) и “известные” порты. С помощью опции `-p` можно явно задать диапазон сканирования.

Команда nmap обладает магической способностью угадывать, какая операционная система установлена на удаленном хосте, путем анализа деталей реализации стека TCP/IP. Иногда ей даже удается определять, какое ПО функционирует на открытом порте. Опции -O и -sV, соответственно, позволяют включать такое поведение. Например:

```
$ nmap -O -sV secure.example.com
Starting Nmap 4.00 ( http://www.insecure.org/nmap/ ) at 2006-05-16 00:01 EDT
Interesting ports on secure.example.com (192.168.1.33):
(The 1670 ports scanned but not shown below are in state: closed)
PORT       STATE    SERVICE  VERSION
22/tcp     open     ssh      OpenSSH 3.6.1p2 (protocol 2.0)
1241/tcp open       ssl      Nessus security scanner
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.4.0 - 2.5.20, Linux 2.5.25 - 2.6.8 or Gentoo 1.2
  Linux 2.4.19 rc1-rc7, Linux 2.6.3 - 2.6.10
Nmap finished: 1 IP address (1 host up) scanned in 8.095 seconds
```

Эта возможность может оказаться весьма полезной при анализе состояния локальной сети. К сожалению, она же является рабочим инструментом хакеров, которые могут определить тип атаки на основании известных слабостей конкретной операционной системы.

Не забывайте также, что большинство администраторов не приветствуют попыток сканирования сети и выяснения ее слабых мест, какими бы благородными ни были мотивы. *Никогда* не запускайте команду nmap в чужой сети без разрешения сетевого администратора.

Nessus: сетевой сканер следующего поколения

Программа Nessus, выпущенная первоначально в 1998 г. Рено Дерезоном (Renaud Deraison), представляет собой мощный и полезный сканер уязвимостей в ПО. Сейчас, когда пишется эта книга, в ее состав входит более 10 000 подключаемых модулей для проверки наличия как локальных, так и удаленных изъянов в системе защиты. Несмотря на то что Nessus представляет собой коммерческий продукт с закрытым кодом, она все равно свободно распространяется, и новые модули для нее выпускаются регулярно. Она является самым популярным и полным сканнером уязвимостей из всех доступных.

Программа Nessus известна как сканер безопасности, который ничего не воспринимает как должное. Вместо того чтобы предполагать, к примеру, будто все Web-серверы работают через порт 80, она ищет Web-серверы на любом порту, а затем анализирует их слабые места. Программа не доверяет даже номеру версии, о котором сообщает сам сервер, проверяя все известные слабые места, чтобы понять, насколько уязвим сервер.

Для первого запуска программы Nessus требуется потратить немало усилий (необходимо установить несколько пакетов, которые не входят в стандартный дистрибутив), но конечный результат того стоит.

Система Nessus состоит из клиента и сервера. Сервер играет роль СУБД, а клиент отвечает за пользовательский интерфейс. Сервер должен работать на платформе UNIX или Linux, зато клиенты существуют для множества платформ.

Одним из главных достоинств программы Nessus является ее модульная структура, позволяющая сторонним разработчикам добавлять собственные модули проверки. Благодаря активности пользователей (и несмотря на недавние лицензионные изменения), Nessus, пожалуй, будет оставаться полезным средством еще долгое время.

Утилита John the Ripper: поиск ненадежных паролей

Один из способов защититься от выбора пользователями ненадежных паролей — попытаться взломать их пароли самостоятельно и затем потребовать от них сменить те пароли, которые удалось раскрыть. John the Ripper — это сложная утилита, которая была разработана Solar Designer и содержит целый ряд алгоритмов для взлома паролей. Она заменила утилиту `crack`, о которой рассказывалось в предыдущих изданиях этой книги.

Несмотря на то что в большинстве версий Linux зашифрованные пароли спрятаны от всеобщего обозрения в файле скрытых паролей, не помешает проверить их стойкость. Знание пользовательского пароля может помочь администратору в обнаружении слабых мест в системе, ведь пользователи стремятся многократно использовать одни и те же пароли. Одинаковый пароль применяется для доступа к другой системе, шифрования файлов в начальном каталоге, доступа к финансовой информации в Internet, и т.д. Стоит ли говорить о том, насколько это неразумно с точки зрения безопасности. Но кому хочется запоминать десять разных паролей?

Несмотря на сложность внутреннего механизма, утилита John the Ripper является чрезвычайно удобной в использовании. Укажите ей файл, который нужно "взломать" (имя такого файла обычно выглядит как `/etc/shadow`), и вы увидите, что за чудеса она творит:

```
$ sudo ./john /etc/shadow
Loaded 25 password hashes with 25 different salts (FreeBSD MD5 [32/32])
password    (bad2)
badpass     (bad1)
```

В данном примере из файла `shadow` было считано целых 25 уникальных паролей. Взламывая пароли, утилита John the Ripper выводит их на экране и затем сохраняет в файле по имени `john.pot`. В выводе, в левом столбце содержится "взломанный" пароль, а в правом — регистрационное имя (login) в круглых скобках. При желании снова отобразить пароли после того, как `john` завершила свой процесс "взлома", следует просто выполнить ту же самую команду с аргументом `-show`.

На момент написания книги самой последней стабильной версией утилиты John the Ripper является 1.7.0.2. Ее можно найти по адресу `www.openwall.com/john`. Поскольку вывод John the Ripper содержит расшифрованные пароли, его следует надежно защищать и удалять сразу же, как только он перестанет быть нужен.

`hosts_access`: управление доступом хостов

Сетевые брандмауэры являются первой линией защиты от доступа с несанкционированных компьютеров, но они не должны быть единственным барьером на этом пути. В Linux имеется два файла — `/etc/hosts.allow` и `/etc/hosts.deny`, — которые используются для ограничения доступа к службам в соответствии с источником сетевых запросов. В файле `hosts.allow` перечисляются хосты, которым разрешено подключаться к той или иной службе, а в файле `hosts.deny` — те, которым запрещается это делать. Однако эти файлы управляют доступом только к службам, которые распознают `hosts_access`, таким как те, которыми управляет `xinetd`, `sshd` и некоторые версии `sendmail`.

В большинстве случаев, лучше быть строгим и разрешать доступ только к самым важным службам и только с определенных компьютеров. Мы рекомендуем запретить

доступ, предоставляемый по умолчанию, за счет добавления в файл `hosts.deny` следующей строки:

```
ALL:ALL
```

а затем разрешить доступ с конкретных компьютеров в файле `hosts.allow`. Ниже для примера показана конфигурация, в которой доступ к SSH разрешается получать только с хостов, находящихся в сетях 192.168/16, а к `sendmail` — из любых хостов:

```
sshd: 192.168.0.0/255.255.0.0
sendmail: ALL
```

Формат записей в любом из этих файлов таков: *служба:компьютер* или *служба:сеть*. Неудачные попытки установки соединения фиксируются в Syslog. Соединения, устанавливаемые с компьютеров, которым не разрешено получать доступ к данной службе, немедленно закрываются.

Большинство дистрибутивов Linux включает файлы `hosts.allow` и `hosts.deny` по умолчанию, но они, как правило, являются пустыми.

Программа Samhain: выявление попыток проникновения на компьютерах

Программа Samhain представляет собой систему выявления непредусмотренных попыток проникновения и мониторинга целостности файлов на компьютерах. Разработал ее и Рейнер Вичман (Rainer Wichmann) из компании Samhain Labs (`la-samhna.de`). Благодаря своему модулярному интерфейсу, она может следить за целостностью файлов, проверять монтируемые файловые системы, выполнять поиск руткитов и др. Например, утилита `samhain` делает очень простым выявление замены злоумышленником установленной копии программы `/bin/login` на копию, которая записывает пароли в секретный файл.

Дизайн программы Samhain с центральной консолью обеспечивает сохранение журнальных данных на надежном компьютере, что помогает поддерживать целостность данных аудита. Отчеты об изменениях на компьютерах могут просматриваться через Web-интерфейс или рассылаться по электронной почте. Журналы Samhain могут пригодиться в качестве криминалистического средства и помочь восстановить последовательность событий после инцидента с безопасностью.

Samhain проверят характеристики компьютеров по базе данных, в которой содержится информация и контрольные суммы файлов, являвшиеся действительными на момент ее создания. Идея состоит в том, чтобы создать базовый образ системы, а затем регулярно сравнивать с ним полученные результаты. Файлы, подверженные частым изменениям при нормальных условиях, могут быть сконфигурированы так, чтобы они не генерировали предупреждений. В случае изменения конфигурации системы или инсталляции новой программы базу данных следует перестраивать, чтобы настоящие проблемы не затерялись в потоке ложных предупреждений.

Вся эта система состоит из трех компонентов. Агент `samhain` запускается на клиентах и отправляет отчеты с данными на сервер регистрации. Демон сервера регистрации, который называется `yule`, принимает соединения от клиентов `samhain` и добавляет данные в сконфигурированные средства регистрации, каковыми обычно являются `Syslog` или база данных. Web-консоль управления, которая называется Beltane, позволяет работать с базовыми образами системы, редактировать конфигурации, а также подписывать конфигурации и базовые образы при помощи цифровой подписи.

Компонентная структура дает этой системе несколько преимуществ по сравнению с традиционными сканнерами целостности файлов. Собираемые данные хранятся на надежном компьютере, поэтому у хакеров гораздо меньше шансов нарушить целостность базы данных. Управление тоже осуществляется легче, потому что информация собирается в одном центральном месте.

Samhain можно сконфигурировать так, чтобы она была чрезвычайно подозрительной. Более того, она умеет делать коварные вещи, например, скрывать себя в списке процессов и присоединять свою базу данных к файлу, который не повреждается от излишнего количества данных, такому как файл JPEG, эффективно пряча базу данных внутри изображения.

Программа Samhain также позволяет создавать при помощи Web-интерфейса криптографические подписи для конфигурационных файлов и баз данных и тем самым упрощать выявление внесения в эти файлы несанкционированных изменений.

Ниже приведен простой пример отчета, сгенерированного Samhain и сохраненного в виде записи в Syslog:

```
# CRIT  : [2006-06-25T19:31:48-0600] msg=<POLICY [ReadOnly] --------T->,
  path=</bin/login>, ctime_old=<[2006-06-26T01:24:34]>, ctime_new=<[2006-
  06-26T01:31:47]>, mtime_old=<[2006-06-26T01:24:34]>, mtime_new=<[2006-
  06-26T01:31:47]>,
  48CA06CC50B857DE77C27956ADE7245B0DF63F6A8A42F5B7
```

Эта запись выглядит немного сложно, но сообщает, что в журнале было зарегистрировано событие типа `CRITICAL` и что атрибуты `ctime` и `mtime` в файле `/bin/login` изменились. А это значит, что пришла пора все проверить.

К сожалению, обслуживание системы Samhain предполагает внушительное количество накладных расходов. Как и любая система, основанная на проверках по базовым данным, система Samhain имеет тенденцию генерировать массу ошибочных жалоб по поводу вполне законных повседневных операций. Конфигурационный файл Samhain позволяет избавиться от жалоб по поводу часто изменяемых файлов, но предвидеть все администраторы не в состоянии. Учтите, что придется частенько держать себя в руках, особенно при обслуживании масштабных установок.

Пакет SELinux (Security-Enhanced Linux)

Как мы увидели, модель защиты в Linux имеет свои недостатки. Разграничительный контроль доступа (концепция доступа к файлам по усмотрению учетной записи с соответствующими разрешениями) является удобным, но не безопасным методом управления доступом к объектам файловой системы. Такая модель традиционно основывается на доверии — доверии в том, что пользователи с правами доступа не имеют злого умысла, что администраторы знают, какие разрешения должны быть у каждого файла в пакете программ, и что программные пакеты сторонних производителей устанавливаются вместе с надежными средствами защиты. Но даже если бы все это было правдой, уязвимость программного обеспечения все равно могла бы оставлять систему незащищенной.

SELinux решает эту проблему путем использования мандатного контроля доступа (mandatory access controls — MAC). При мандатном контроле доступа пользователи не имеют официального права управлять доступом к объектам. Вместо этого, администратор определяет общесистемные политики доступа. Хорошо реализованная политика MAC заключается в следовании принципу “наименьшего уровня привилегий”, т.е. раз-

решении доступа только тогда, когда это действительно необходимо, во многом подобно тому, как правильно настроенный брандмауэр позволят проходить только специально распознаваемым службам и клиентам. MAC может предотвращать нарушение защиты системы ПО с уязвимыми местами в коде выполнения (например, переполнениями буфера), ограничивая масштаб бреши несколькими конкретными, требующимися этому ПО ресурсами.

SELinux — это проект Агентства национальной безопасности США, который свободно распространяется с конца 2000 г. Он был интегрирован в ядра Linux серии 2.6. Отдельными дистрибутивами, однако, он адаптируется вяло, за исключением разве что таких дистрибутивов как Red Hat Enterprise Linux и Fedora.

Разработка политик является сложной темой. Одна компания точно предлагает трехдневные курсы по политиками SELinux. Для защиты нового демона, например, в политике должны быть тщательно перечислены все файлы, каталоги и другие объекты, доступ к которым может быть необходим этому процессу. В случае сложного программного обеспечения, такого как программа `sendmail` или Apache-утилита `httpd`, выполнить эту задачу может быть очень трудно.

К счастью, многие общие политики доступны в Internet. Они легко устанавливаются и конфигурируются. Полнофункциональный редактор, упрощающий применение политик, доступен по адресу `seedit.sourceforge.net`.

SELinux присутствует в дистрибутиве RHEL, начиная с версии 4, а в дистрибутиве Fedora, начиная с версии Core2. Стандартная установка Fedora и Red Hat Enterprise Linux на самом деле предполагает активизацию некоторых защитных политик SELinux с самого начала.

За конфигурацию SELinux отвечает файл `/etc/selinux/config`. В этом файле есть две таких интересных строки:

```
SELINUX=enforcing
SELINUXTYPE=targeted
```

В первой строке могут указываться три следующих значения: `enforcing`, `permissive` или `disabled`. Установка значения `enforcing` гарантирует применение загруженной политики и предотвращение попыток ее нарушения. В случае установки значения `permissive` попытки нарушения допускаются, но при этом регистрируются в системе Syslog. А в случае установки значения `disabled` программа SELinux вообще полностью отключается.

В строке `SELINUXTYPE` указывается тип применяемой политики. В Red Hat и Fedora используются политики двух типов: политика `targeted`, которая подразумевает дополнительную защиту для защищенных Red Hat демонов[8], и политика `strict`, которая подразумевают защиту всей системы. Хотя политика `strict` доступна, Red Hat ее не поддерживает: ограничения являются такими жесткими, что системой становится очень трудно пользоваться. Политика `targeted` обеспечивает защиту для важных сетевых демонов, не влияя (по крайней мере, теоретически) на общее удобство пользования системой. Но даже политика `targeted` не идеальна. При наличии проблем с новым установленным ПО нужно проверить файл `/var/log/messages` на наличие ошибок SELinux.

В SUSE применяется ее собственная реализация MAC, которая называется *AppArmor* и не включает SELinux.

[8] К таким защищенным демонам относятся: `httpd`, `dhcpd`, `mailman`, `named`, `portmap`, `nscd`, `ntpd`, `mysqld`, `postgres`, `squid`, `winbindd` и `ypbind`.

Пакетами SELinux для Debian и Ubuntu занимается Рассел Коукер (Russel Coker), сотрудник Red Hat, создавший политики `strict` и `targeted`.

20.11. Системы криптографической защиты

Многие часто используемые в Linux протоколы были созданы еще до широкого распространения Internet и до изобретения современных криптографических систем. При проектировании многих протоколов о безопасности просто не думали. Там же, где она якобы учитывалась, реализованные механизмы дискредитируются передачей паролей в незашифрованном виде или отсутствием проверки на надежность компьютера или порта, от которого поступил пакет.

Сегодня эти протоколы эксплуатируются в кишащих хакерами и прочей нечистью глобальных сетях, где весь трафик открыт для прослушивания. Более того, кто угодно может вмешаться в сетевой диалог. Как удостовериться в личности собеседника?

Криптография — это решение многих проблем. Давным-давно существовала возможность шифровать сообщения так, чтобы даже в случае перехвата их нельзя было прочитать, но это лишь одно из чудес криптографии. Такие разработки, как шифрование с открытым ключом и защитное хеширование, сделали возможным построение криптосистем, удовлетворяющих самым строгим требованиям[9].

К сожалению, все эти математические открытия не удалось преобразовать в надежное, полезное программное обеспечение, которое было бы повсеместно распространено и удобно в применении. Разработчиков программных криптографических систем больше интересовало доказательство их правильности и абсолютной безопасности, нежели то, имеет ли такая система реальный практический интерес. Почти все современные программы данного направления выглядят слишком заумными, поэтому не удивительно, что при первой же возможности пользователи с радостью избавляются от них. Сегодня практической криптографией занимаются фанатики, параноидальные сторонники теории всемирного заговора, а также те, кому административная политика их организации не оставляет другого выбора.

Непонятно, появится ли что-нибудь более разумное в данной области в ближайшие несколько лет. А пока мы познакомимся с тем, что сегодня имеется на рынке.

Kerberos: унифицированный подход к сетевой безопасности

Система Kerberos, разработанная в Массачусетсском технологическом институте, ориентирована на решение задач, связанных с обеспечением безопасности компьютерных сетей. Kerberos — это система аутентификации, предоставляющая "гарантию" того, что пользователи и служебные программы действительно являются теми, за кого себя выдают. Никаких дополнительных проверок и шифрования передаваемых данных не предусмотрено.

Используя алгоритм DES, Kerberos создает вложенные наборы идентификаторов, называемых *билетами*, или *мандатами*. Мандаты передаются по сети с целью подтвер-

[9] Тем, кого интересуют вопросы криптографии, рекомендуем обратиться к двум великолепным источникам: ресурсу "RSA Labs' Frequently Asked Questions about Today's Cryptography" (`www.rsasecurity.com/rsalabs/faq`) и книге *Crypto* ("Криптография") Стефена Леви (Stephen Levy), которая является прекрасным путеводителем по истории развития криптографических технологий.

ждения личности пользователя и предоставления ему доступа к сетевым службам. В каждой организации, где используется Kerberos, должен выделяться хотя бы один физически защищенный компьютер (называемый сервером аутентификации) для выполнения демона Kerberos. Этот демон выдает мандаты пользователям и служебным программам, требующим аутентификации, на основании предъявляемых ими "удостоверений", в частности паролей.

По сути, Kerberos улучшает традиционную схему парольной защиты Linux всего двумя способами: пароли никогда не передаются по сети в незашифрованном виде и пользователи избавляются от необходимости многократно вводить пароли.

Сообщество пользователей Kerberos может похвастаться одним из наиболее толковых и оригинальных документов, посвященных криптосистемам: "Designing an Authentication System: a Dialogue in Four Scene" (проектирование системы аутентификации: диалог в четырех актах) авторства Билла Брайанта (Bill Bryant). Его будет интересно почитать любому, кто интересуется темой криптографии. Документ можно найти по адресу

```
web.mit.edu/kerberos/www/dialogue.html
```

Имеется также хороший список часто задаваемых вопросов:

```
www.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html
```

Лучше использовать Kerberos, чем вообще отказаться от средств защиты. К сожалению, Kerberos нельзя назвать полностью безопасной системой, а ее инсталляция и запуск — не самое приятное занятие. В то же время она не заменяет собой ни одну из программ, описанных в данной главе. По нашему мнению, большинство организаций благополучно может обойтись и без нее. Регулярная проверка системы и применение системы шифрования удаленных соединений, в частности SSH (см. ниже), позволят обеспечить более чем адекватный уровень безопасности пользователей.

К сожалению (и, возможно, как и следовало ожидать), система Kerberos, распространяемая как часть Windows 2000, использует патентованные, недокументированные расширения протоколов. Как следствие, она плохо взаимодействует с традиционной версией системы.

PGP: высокая конфиденциальность

Пакет PGP (Pretty Good Privacy), написанный Филипом Циммерманом (Philip Zimmermann), содержит криптографические утилиты, ориентированные в основном на безопасность систем электронной почты. Посредством PGP можно шифровать данные, генерировать сигнатуры и проверять источник происхождения файлов и сообщений.

Попытки то отрегулировать, то остановить распространение пакета привели к появлению разнообразных его вариантов. Например, компания McAfee поставляет PGP как часть своего серверного продукта E-Business, а корпорация PGP использует PGP в целом ряде своих связанных с шифрованием продуктов. Проект GNU предоставляет замечательную, бесплатную и широко применяемую реализацию PGP, известную как GnuPG и доступную по адресу `www.gnupg.org`. В США используется версия, одобренная правительством. Международная версия с усиленным алгоритмом шифрования доступна на Web-сайте `www.pgpi.org`.

PGP — наиболее популярная криптографическая программа. К сожалению, для того чтобы работать с ней в UNIX/Linux, нужно быть специалистом в области криптографии. С другой стороны, к пакету прилагается 88-страничный (всего-то) учебник по криптографии, так что кто хочет, тот всегда сможет разобраться. Если говорить в общем, то

пакет PGP может оказаться полезным, но мы не рекомендуем заставлять пользователей работать с ним из-за нетривиальности процесса обучения. Гораздо удобнее использовать Windows-версию PGP, чем команду `pgp` с ее 38 различными режимами работы.

Программные пакеты, распространяемые через Internet, часто содержат файл сигнатуры PGP, подтверждающий их подлинность и целостность. Но проверить эти сигнатуры не так-то легко пользователям, не являющимся фанатиками PGP. Не потому, что процесс проверки сложен, а потому, что при использовании PGP истинная безопасность достигается только путем создания персональной коллекции открытых ключей тех людей, которые были проверены напрямую. Загрузка дистрибутива с открытым ключом и файлом сигнатуры безопасна примерно в той же степени, что и загрузка одного дистрибутива.

SSH: безопасный интерпретатор команд

Пакет SSH (Secure Shell) является безопасной заменой утилит `rlogin`, `rcp` и `telnet`. Он использует криптографическую аутентификацию для подтверждения личности пользователя и шифрует любые соединения между двумя компьютерами. Протокол, используемый пакетом, разрабатывался для противодействия целому ряду возможных атак. Этот протокол был описан в документах RFC с 4250 по 4256 и сейчас проходит процесс стандартизации в IETF.

Пакет SSH трансформировался из свободно распространяемой открытой системы (SSH1) в коммерческий продукт с немного иным (более надежным) протоколом (SSH2). К счастью, сообщество разработчиков открытых систем взяло ситуацию под контроль и выпустило замечательный пакет OpenSSH, в котором сейчас реализованы оба протокола.

Основными компонентами пакета SSH являются серверный демон `sshd` и две утилиты пользовательского уровня: `ssh`, предназначенная для удаленной регистрации, и `scp` — для копирования файлов. Среди остальных компонентов отметим команду `ssh-keygen`, генерирующую пары открытых ключей, и несколько утилит, необходимых для поддержки безопасных серверов X Window.

Демон `sshd` способен аутентифицировать пользователей различными способами. Выбор остается за администратором.

- **Метод А**. Если имя удаленного компьютера, с которого регистрируется пользователь, указано в файле `~/.rhosts`, `~/.shosts`, `/etc/hosts.equiv` или `/etc/shosts.equiv`, то пользователь автоматически получает доступ в систему без проверки пароля. Подобная схема моделирует работу старого демона `rlogind` и, по нашему мнению, неприемлема для широкого применения.
- **Метод Б**. В дополнение к методу А демон `sshd` может применять шифрование с открытым ключом для проверки адреса удаленного компьютера. Чтобы это произошло, открытый ключ компьютера (генерируется при инсталляции пакета) должен находиться в файле `/etc/ssh_known_hosts` локального хоста или в файле `~/.ssh/known_hosts` пользователя. Если удаленный компьютер предоставляет соответствующий секретный ключ (обычно хранится в файле `/etc/ssh_host_key`, который недоступен для чтения рядовым пользователям), то пользователю разрешается войти в систему без проверки пароля. Мы полагаем, что данный метод недостаточно безопасен. Если удаленный компьютер взломан, локальная система также окажется под угрозой.

- **Метод В**. Демон `sshd` может применять шифрование с открытым ключом для идентификации самого пользователя. На этапе регистрации пользователь должен иметь доступ к файлу своего секретного ключа и предоставить пароль для его дешифровки. Этот метод наиболее безопасный, но он утомляет пользователей. Он также делает невозможной регистрацию в системе мобильных пользователей (разве что они носят с собой копию файла секретного ключа, например зашифрованного ключа USB). Решив использовать пары ключей, при выявлении проблем почаще пользуйтесь командой `ssh -v`.
- **Метод Г**. Наконец, демон `sshd` может просто попросить пользователя ввести регистрационный пароль. Это делает утилиту `ssh` похожей на утилиту `telnet`, за исключением того, что пароль и все данные, передаваемые в ходе сеанса, подвергаются шифрованию. Основной недостаток этого метода заключается в том, что пароли могут быть относительно слабыми, если не усилить их защиту, и что есть готовые программы (наподобие программы John the Ripper), которые могут их взломать. Тем не менее, этот метод, как правило, больше всего подходит для повседневного использования.

Правила аутентификации задаются в файле `/etc/sshd_config`. Он уже заполнен разного рода конфигурационным "мусором", большую часть которого можно проигнорировать. Параметры, касающиеся аутентификации, перечислены в табл. 20.1.

Таблица 20.1. Параметры аутентификации, задаваемые в файле `/etc/sshd_config`

Параметр	Метод[а]	По умолчанию	Назначение, когда включен
`RhostsAuthentication`	А	`no`	Разрешает регистрацию через файлы `~/.shosts`, `/etc/shosts.equiv` и т.п.
`RhostsRSAAuthentication`	Б	`yes`	Разрешает регистрацию через файл `~/.shosts` и др., но также требует от компьютера предоставить ключ
`IgnoreRhosts`	А, Б	`no`	Задает игнорирование файлов `~/.rhosts` и `hosts.equiv`[б]
`IgnoreRootRhosts`	А, Б	`no`[в]	Запрещает аутентификацию пользователя `root` через файлы `.rhosts` и `.shosts`
`RSAAuthentication`	В	`yes`	Разрешает аутентификацию пользователей по методу шифрования с открытым ключом
`PasswordAuthentication`	Г	`yes`	Разрешает использование обычных регистрационных паролей

[а] Методы аутентификации, к которым относится параметр.

[б] Файлы `~/.shosts` и `shosts.equiv` продолжают использоваться.

[в] По умолчанию равен значению параметра `IgnoreRhosts`.

Мы рекомендуем использовать конфигурацию, которая разрешает пользоваться методами В и Г и запрещает пользоваться методами А и Б:

```
RhostsAuthentication no
RhostsRSAAuthentication no
RSAAuthentication yes
PasswordAuthentication yes
```

Никогда не следует разрешать учетной записи `root` регистрироваться удаленно. Доступ суперпользователя должен получаться путем использования `sudo`. Для стимуляции такого поведения можно воспользоваться следующей опцией:

```
PermitRootLogin no
```

Одноразовые пароли

Программы, основанные на применении метода грубой силы, такие как программа John the Ripper, заставляют обращать внимание на ненадежность статических паролей. Даже при наличии инструментов вроде PAM и механизма устаревания паролей, пользователи все равно выбирают легко запоминающиеся, но ненадежные пароли, потом записывают их на клейких бумажках и делятся ими со своими коллегами по работе. Одноразовые пароли позволяют решить эту проблему, вынуждая пользователей вводить уникальный пароль при каждой регистрации в системе. Из-за того, что пароль постоянно изменяется, пропадает смысл использовать программы взлома.

В наши дни одноразовые пароли наиболее часто встречаются в коммерческих системах защиты. Многие поставщики выпускают системы такого типа, имеющие вид небольшой карточки с жидкокристаллическим дисплеем. Эти карточки динамически генерируют пользовательские пароли.

Программа Stunnel

Программа Stunnel, созданная Михалем Тройнара (Michal Trojnara) представляет собой распространяемый с открытым исходным кодом пакет, который шифрует произвольные TCP-соединения. Она использует протокол SSL для создания сквозных “туннелей”, по которым потом передает данные незашифрованной службе и обратно. Известно, что она хорошо работает с такими небезопасными службами как Telnet, IMAP и POP.

Демон `stunnel` запускается как на клиентской, так и на серверной системе. Локальный демон `stunnel` обычно принимает соединения, поступающие через традиционный порт службы (например, порт 25, если говорить об SMTP), и направляет их через SSL демону `stunnel`, находящемуся на удаленном компьютере. Удаленный демон `stunnel` принимает соединение, расшифровывает входящие данные и направляет их на удаленный порт, который прослушивает сервер. Такая система позволяет незашифрованным службам пользоваться такими преимуществами шифрования, как конфиденциальность и целостность данных, без внесения каких-либо программных изменений. Программное обеспечение клиента нужно только сконфигурировать так, чтобы оно выполняло поиск служб на локальной системе, а не на сервере, который в конечном итоге будет их предоставлять.

Telnet может послужить хорошим примером, потому что состоит из простого демона, который прослушивает один единственный порт. Чтобы настроить соединение Telnet при помощи `stunnelfy`, сначала нужно создать SSL-сертификат. Stunnel не зависит ни от каких библиотек SSL, так что никакая ее основанная на стандартах реализация тоже не будет; нам нравится OpenSSL. Чтобы сгенерировать сертификат, нужно сделать следующее:

```
server# openssl req -new -x509 -days 365 -nodes -out stunnel.pem -keyout
        stunnel.pem
Generating a 1024 bit RSA private key
.++++++
..............................++++++
writing new private key to 'stunnel.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
Country Name (2 letter code) [GB]:US
State or Province Name (full name) [Berkshire]:Colorado
Locality Name (eg, city) [Newbury]:Boulder
Organization Name (eg, company) [My Company Ltd]:Booklab, Inc.
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
        server.example.com
Email Address []:

Запись нового секретного ключа в 'stunnel.pem'
-----
Далее последует запрос на ввод информации, которая будет включена в
сертификат.
Такая информация называется "отличительным именем" или DN (Distinguished Name).
Строк для ввода будет всего несколько и некоторые из них можно будет
оставить пустыми.
В некоторых строках уже будет указано значение по умолчанию,
В случае ввода '.' строка будет оставлена незаполненной.
Страна (код из двух букв) [GB]:US
Штат или область (полностью) [Berkshire]:Colorado
Местонахождение (например, город ) [Newbury]:Boulder
Название организации (например, компания) [My Company Ltd]:Booklab, Inc.
Отдел []:
Общее имя (например, ВАШЕ имя или имя главного хоста сервера) []:
        server.example.com
Электронный адрес[]:
```

Эта команда создает самоподписывающийся и не имеющий идентификационной фразы сертификат. Хотя отсутствие идентификационной фразы является удобным (потому что позволяет не присутствовать администратору лично при каждом перезапуске `stunnel` для ввода этой фразы), оно также представляет угрозу безопасности. Поэтому необходимо обязательно защитить файл сертификата очень строго ограниченными разрешениями.

Далее нужно определить конфигурацию как для серверного, так и для клиентского демона `stunnel`. Стандартным конфигурационным файлом является файл `/etc/stunnel/stunnel.conf`, но вы можете создать несколько конфигураций, если хотите иметь несколько туннелей:

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
```

```
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = no

[telnets]
accept = 992
connect = 23
```

В этой конфигурации следует обратить внимание на два важных момента. Во-первых, обратите внимание на оператор `chroot`, который ограничивает процесс `stunnel` каталогом `/var/run/stunnel`. Пути для дополнительных файлов могут указываться либо в обычном системном пространстве имен, либо в пространстве имен `chroot`, в зависимости от точки, в которой они открываются. Здесь файл `stunnel.pid` фактически находится в каталоге `/var/run/stunnel`.

В разделе `[telnetd]` содержится два оператора: оператор `accept` указывает `stunnel` принимать соединения через порт 992, а оператор `connect` — передавать эти соединения через порт 23, который прослушивает служба Telnet.

Конфигурация клиента выглядит очень похоже:

```
cert = /etc/stunnel/stunnel.pem
chroot = /var/run/stunnel/
pid = /stunnel.pid
setuid = nobody
setgid = nobody
debug = 7
output = /var/log/stunnel.log
client = yes

[telnets]
accept = 23
connect = server.example.com:992
```

Здесь несколько директив выполняет операции, являющиеся обратными по отношению к серверной конфигурации. В частности, оператор `client = yes` указывает программе инициировать соединения `stunnel`, а не принимать их. Локальный демон `stunnel` должен принимать соединения через порт 23, а к серверу подключаться — через порт 992. Имя хоста в директиве `connect` должно соответствовать тому, которое было указано при создании сертификата.

Как клиентский, так и серверный демон `stunnel` может запускаться без аргументов командной строки. Если выполнить проверку с помощью команды `netstat-an`, вывод покажет, что серверный демон `stunnel` прослушивает порт 992, а клиентский — порт 23.

Чтобы получить доступ к туннелю, пользователь должен просто отправить запрос к службе `telnet` локальному компьютеру:

```
client# telnet localhost 23
Trying 127.0.0.1...
Connected to localhost (127.0.0.1).
Escape character is '^]'.
Red Hat Enterprise Linux WS release 4 (Nahant Update 2)
Kernel 2.6.9-5.EL on an i686
login:
```

Теперь пользователь может безопасно войти в систему, не боясь кражи пароля. Бдительный администратор будет с осторожностью использовать TCP-упаковщики

для ограничения соединений на клиенте только устанавливаемыми через локальный интерфейс: ведь главное сделать так, чтобы никакой человек из внешнего мира не мог безопасно подключаться к серверу через Telnet! `stunnel` — это одна из нескольких программ, которые имеют встроенную поддержку для упаковщиков и не требуют ограничивать доступ при помощи `tcpd`. Все необходимые инструкции можно найти по адресу `www.stunnel.org`.

20.12. Брандмауэры

Помимо защиты отдельных компьютеров можно предпринять меры для обеспечения безопасности на сетевом уровне. Основным инструментом сетевой защиты является брандмауэр. Есть три категории брандмауэров в зависимости от типа фильтрации: на уровне пакетов, на уровне служб и с учетом состояния.

Фильтрация пакетов

Брандмауэр, фильтрующий пакеты, ограничивает трафик через Internet-шлюз (или через внутренний шлюз, соединяющий домены в пределах организации), основываясь на информации из заголовков пакетов. Администратор указывает, какие целевые адреса, номера портов и типы протоколов являются допустимыми, а брандмауэр просто отбрасывает (в некоторых случаях также регистрирует) все пакеты, которые не соответствуют заданным критериям.

Фильтрацию пакетов выполняют и специализированные маршрутизаторы (например, выпускаемые компанией Cisco). Возможна также программная фильтрация; все зависит от компьютера, выполняющего функции шлюза, и его конфигурации. В целом фильтрующие брандмауэры значительно повышают безопасность системы, не оказывая существенного влияния на производительность и не требуя сложной настройки.

В Linux имеется программный фильтр пакетов (см. раздел 20.13). Можно также приобрести коммерческую программу для этих целей. Все они позволят обеспечить разумную защиту домашнего компьютера или небольшого офиса. Но прочитайте еще раз начало главы, прежде чем рассматривать Linux-систему в качестве корпоративного брандмауэра[10]. Это как раз тот случай, когда действительно стоит вложить деньги в покупку специализированного сетевого оборудования, например брандмауэра PIX компании Cisco.

Принципы фильтрации служб

Большинству “известных” служб назначается сетевой порт в файле `/etc/services` (или его системно-зависимом эквиваленте). Демоны, которые реализуют соответствующие службы, постоянно опрашивают порты, ожидая поступления запросов на подключение со стороны удаленных компьютеров[11]. В основном такие порты являются “привилегированными”. Это значит, что их номера находятся в диапазоне от 1 до 1023 и что

[10] Мы также предполагаем, что пользователям не приходят в голову глупые идеи наподобие того, как сделать Windows платформой для брандмауэра. Неужели при мысли о Windows у кого-то возникает чувство безопасности? Давайте не заниматься ерундой: Windows — для настольных компьютеров.

[11] Во многих случаях задачу ожидания берет на себя демон `xinetd`. Подробнее об этом демоне будет рассказываться в разделе 29.3.

порты могут использоваться только процессами, работающими от имени пользователя root. Порты с номерами 1024 и выше являются непривилегированными.

Фильтрация на уровне служб основана на предположении, что клиент (компьютер, инициирующий соединение по протоколу TCP или UDP) будет использовать непривилегированный порт для установления соединения с привилегированным портом сервера. Например, если компьютеру с адресом 192.108.21.200 нужно разрешить только входящие SMTP-соединения, то следует установить фильтр, который позволит посылать TCP-пакеты по этому адресу через порт 25 и отправлять TCP-пакеты с этого адреса через любой порт[12]. Способ инсталляции такого фильтра будет зависеть от типа применяемого маршрутизатора.

Некоторые службы, например FTP, еще больше усложняют эту головоломку. При пересылке файла по протоколу FTP устанавливается на самом деле два соединения: одно — для команд, другое — для данных. Клиент инициирует командное соединение, а сервер — информационное. Следовательно, если необходимо получать файлы из Internet по протоколу FTP, нужно разрешить вход в систему через все непривилегированные TCP-порты, так как неизвестно, какой из них будет использован для получения данных.

📖 Подробнее о конфигурировании FTP-сервера будет рассказываться в разделе 21.6.

Это сводит на нет преимущества фильтрации пакетов, поскольку некоторые общеизвестные службы, славящиеся своей незащищенностью, работают с непривилегированными портами (например, X11 — порт 6000). Кроме того, любознательные пользователи получают возможность запускать собственные служебные программы (например, сервер `telnet` на нестандартном и непривилегированном порту), к которым они и их друзья могут обращаться из Internet.

Одним из наиболее популярных решений проблемы FTP является использование протокола передачи файлов с применением SSH (SFTP). Этот протокол на текущий момент доступен в Internet пока только в черновом варианте, но уже широко используется и является довольно зрелым. Он очень часто используется в качестве одного из компонентов протокола SSH, обеспечивая для него аутентификацию и шифрование. В отличие от FTP, SFTP использует один единственный порт как для команд, так и для данных, эффективно устраняя парадокс с фильтрацией пакетов. Существует целый ряд различных реализаций SFTP. Нам очень понравилось, как работает SFTP-клиент командной строки, поставляемый вместе с OpenSSH.

Если протокол FTP все-таки нужно применять обязательно, самым разумным подходом будет разрешить выход по протоколу FTP во внешний мир только с одного единственного изолированного компьютера. Пользователи смогут регистрироваться на нем, когда им понадобится выполнять сетевые операции, запрещенные во внутренней сети. Поскольку копирование всех пользовательских учетных записей на FTP-сервер противоречит сути административного деления, можно создавать учетные записи только по заявкам. Естественно, на FTP-сервере должен быть инсталлирован полный комплект средств защиты.

Самый безопасный способ использования фильтра пакетов — начать с конфигурации, в которой не допускается ничего, кроме входящего SMTP- или SSH-трафика. Затем можно постепенно ослаблять правила фильтрации по мере того, как будет обнаруживаться, что отдельные полезные службы не работают.

[12] Порт 25 — это порт SMTP, определенный в файле `/etc/services`.

В некоторых заботящихся о безопасности организациях используется двухступенчатая фильтрация. Один фильтр в этой схеме подключен к Internet как шлюз, а второй находится между этим шлюзом и остальной частью локальной сети. Идея состоит в том, чтобы оставить наружный шлюз относительно открытым, а внутренний — сделать очень консервативным. Если промежуточные компьютеры административно отделены от остальной части сети, появляется возможность реализовывать различные службы Internet с меньшим риском. Частично защищенную сеть обычно называют *демилитаризованной зоной* (demilitarized zone — DMZ).

Фильтрация на уровне служб

Брандмауэры, осуществляющие фильтрацию на уровне служб, перехватывают входящие и исходящие запросы на подключение и устанавливают собственные соединения с демонами в пределах сети, играя роль преграждающего затвора либо посредника между двумя мирами. По своей структуре такие брандмауэры являются менее гибкими и быстродействующими, чем обычные фильтры пакетов. Они должны иметь специальный модуль декодирования и перенаправления пакетов для каждого протокола, трафик которого проходит через брандмауэр. В начале 90-х гг. это было относительно несложно сделать, поскольку широко использовалось лишь небольшое число протоколов. Сегодня, путешествуя в глобальной сети, пользователь за один час может задействовать несколько десятков протоколов. Поэтому брандмауэры, осуществляющие фильтрацию на уровне служб, относительно непопулярны в организациях, где основная среда взаимодействия — Internet.

Фильтрация с учетом состояния

Принцип, лежащий в основе анализирующих брандмауэров, можно объяснить с помощью такой аналогии: при внимательном прослушивании разговоров (на всех языках) в толпе аэропорта можно узнать, не планирует ли кто-нибудь пронести бомбу в самолет. Такие брандмауэры проверяют проходящий через них трафик и сравнивают реальную сетевую активность с той, которая, по их мнению, должна происходить. Например, если в пакетах, передаваемых в процессе обмена FTP-командами, указывается порт, через который должно быть установлено информационное соединение, брандмауэр обязан убедиться в том, что запрос поступит именно на этот порт. Попытки удаленных хостов подключиться к другим портам будут считаться ошибочными и потому будут игнорироваться.

К сожалению, суровая реальность вносит свои коррективы. Нереально отслеживать “состояние” сетевых соединений для тысяч компьютеров и сотен протоколов, как нереально пытаться подслушать каждый разговор в переполненном аэропорту. Возможно, когда-нибудь, когда процессорные мощности и объемы оперативной памяти возрастут на порядок, этот замысел и воплотится в жизнь.

Так что же нам пытаются продать под маркой “анализирующие брандмауэры”? Такие системы либо контролируют очень ограниченное число соединений и протоколов, либо ищут конкретные “неправильные” ситуации. Нельзя сказать, что это плохо. Полезна любая технология, позволяющая выявлять аномалии трафика. Но в данном конкретном случае нужно помнить о том, что большинство рекламных заявлений такого рода так и остается рекламой.

Насколько безопасны брандмауэры?

Фильтрация пакетов не должна быть основным средством защиты от хакеров. Это всего лишь вспомогательная мера. При наличии брандмауэра порой создается ложное впечатление, будто уровень безопасности является исключительно высоким. Если, доверившись брандмауэру, ослабить бдительность на других "рубежах", это *отрицательно* скажется на защищенности информации.

Каждый компьютер необходимо защищать индивидуально и регулярно проверять, пользуясь такими средствами, как `xinetd`, Nmap, Nessus и Samhain. Следует также обучать пользователей правилам безопасной работы в системе.

В идеале локальные пользователи должны иметь возможность подключаться к любой службе Internet, но доступ к локальным службам со стороны Internet-хостов должен быть ограничен. Например, к серверу архивов может быть разрешен FTP-доступ, а к почтовому серверу — доступ только по протоколу SMTP.

Чтобы Internet-канал работал с максимальной эффективностью, советуем все же обратить основное внимание на удобство работы и доступность. В конце концов, сеть становится безопасной благодаря бдительности системного администратора, а не благодаря брандмауэру.

20.13. Поддержка брандмауэров в Linux: IP-таблицы

Обычно мы не рекомендуем использовать компьютер, работающий под управлением Linux (а также UNIX или Windows), в качестве брандмауэра, поскольку в целом операционная система не обеспечивает надлежащий уровень безопасности. Оптимальными для этих целей являются специализированные устройства маршрутизации и фильтрации пакетов (например, Cisco PIX)[13]. Но защищенная система Linux является замечательным заменителем для организаций, бюджет которых не предусматривает покупку дорогостоящего оборудования для брандмауэра.

Устанавливая Linux-систему в качестве брандмауэра, убедитесь, что она включает самые последние обновления и "заплаты", касающиеся брешей в системе защиты. Компьютер, функционирующий в качестве брандмауэра, — идеальное место для практической проверки всех рекомендаций, изложенных в этой главе. (В разделе 20.12 собраны сведения, касающиеся брандмауэров в целом. Читателям, не знакомым с концепцией брандмауэра, советуем предварительно прочитать приведенный там материал.)

В ядре Linux версии 2.4 появился новый механизм фильтрации пакетов: Netfilter. Контролирующая его работу команда `iptables` является "наследницей" более старой команды `ipchains`, которая использовалась в ядрах версии 2.2. В команде `iptables` применяется концепция упорядоченной *цепочки* правил, согласно которым проверяются сетевые пакеты. Наборы цепочек образуют *таблицы*, предназначенные для обработки определенных видов трафика.

К примеру, стандартная таблица называется `filter`. Цепочки правил этой таблицы используются для фильтрации сетевых пакетов. В таблице `filter` содержатся три стандартные цепочки. Каждый пакет, обрабатываемый ядром, поступает на проверку в одну цепочку. Правила в цепочке `FORWARD` применяются ко всем пакетам, которые поступают в один интерфейс и перенаправляются в другой. Правила в цепочках `INPUT` и

[13] При этом во многих ориентированных на потребителя сетевых устройствах, таких как маршрутизаторы компании Linksys, система Linux и `iptables` используются прямо в их ядре.

OUTPUT применяются к пакетам, адресованным локальному компьютеру или исходящим от него, соответственно. Обычно этих трех цепочек вполне достаточно для фильтрации данных между двумя интерфейсами. В случае необходимости можно определять более сложные конфигурации.

Помимо таблицы filter существуют также таблицы nat и mangle. В первой содержатся цепочки правил, предназначенные для управления системой NAT (Network Address Translation — трансляция сетевых адресов). Подробнее об этой системе рассказывалось в разделе 12.4, а в разделе 12.12 приводился пример использования таблицы NAT. В этом разделе мы воспользуемся цепочкой PREROUTING данной таблицы для фильтрации пакетов с поддельными IP-адресами.

Таблица mangle содержит цепочки, предназначенные для модификации содержимого сетевых пакетов вне контекста системы NAT или механизма фильтрации пакетов. Эта таблица редко применяется в крупных системах, поэтому мы не будем ее рассматривать.

В каждом правиле цепочки имеется директива, определяющая, что следует делать с пакетами, подчиняющимися этому правилу. Если пакет прошел проверку, его судьба предрешена, и другие проверки не выполняются. В качестве директивы разрешается указывать имя другой цепочки.

Правила таблицы filter могут содержать следующие директивы: ACCEPT, DROP, REJECT, LOG, MIRROR, QUEUE, REDIRECT, RETURN и ULOG. Директива ACCEPT разрешает пакету следовать своим маршрутом. Директивы DROP и REJECT запрещают пропускать пакет, но первая приводит к "безмолвному" удалению пакета, а вторая — к выдаче ICMP-сообщения об ошибке. Директива LOG позволяет следить за тем, какие правила применяются к пакетам, а директива ULOG включает режим расширенной регистрации.

Директива REDIRECT перенаправляет пакеты прокси-серверу. Она удобна, когда весь трафик Web-сайта должен проходить через кэширующую программу, такую как Squid. Директива RETURN объявляет конец пользовательской цепочки. Директива MIRROR меняет местами IP-адреса отправителя и получателя перед отправкой пакета. Наконец, директива QUEUE передает пакеты локальным программам пользователя через модуль ядра.

В Linux брандмауэр обычно реализуется в виде последовательности команд iptables, содержащихся в одном из сценариев запуска системы. Отдельные команды iptables, как правило, имеют одну из следующих форм:

```
iptables -F имя_цепочки
iptables -P имя_цепочки директива
iptables -A имя_цепочки -i интерфейс -j директива
```

В первом случае (-F) из цепочки удаляются все предыдущие правила. Во втором случае (-P) задается стандартная директива цепочки. Мы рекомендуем по умолчанию использовать директиву DROP. В третьем случае (-A) заданное правило добавляется к цепочке. Если не указан флаг -t, команда применяется к цепочкам таблицы filter. Опция -i определяет интерфейс, для которого это правило действует. Опция -j задает директиву. Команда iptables понимает также ряд других опций (табл. 20.2).

Таблица 20.2. Дополнительные опции фильтрации команды `iptables`

Опция	Назначение
`-p` *протокол*	Соответствие протоколу: `tcp`, `udp` или `icmp`
`-s` *исходный_адрес*	Соответствие исходному IP-адресу хоста или сети (допускается нотация CIDR)
`-d` *целевой_адрес*	Соответствие целевому IP-адресу хоста или сети
`--sport` *номер_порта*	Соответствие номеру исходного порта (обратите внимание на двойной дефис)
`--dport` *номер_порта*	Соответствие номеру целевого порта (обратите внимание на двойной дефис)
`--icmp_type` *тип*	Соответствие типу ICMP-сообщения (обратите внимание на двойной дефис)
`!`	Инверсия смысла опции
`-t`	Задает таблицу, к которой применяется команда (по умолчанию — `filter`)

Ниже мы детально разберем конкретный пример. Предполагается, что интерфейс `ppp0` подключен к Internet, а интерфейс `eth0` — к локальной сети. IP-адрес интерфейса `ppp0` равен 128.138.101.4, а интерфейса `eth0` — 10.1.1.1. В обоих случаях маска сети равна 255.255.255.0. В этом примере выполняется фильтрация пакетов без учета состояния для защиты Web-сервера с IP-адресом 10.1.1.2. Это стандартный метод защиты Web-серверов. В конце примера будет продемонстрирована фильтрация пакетов с учетом состояния для защиты настольных компьютеров.

Прежде чем использовать `iptables` в качестве брандмауэра, следует включить режим перенаправления IP-пакетов и убедиться в том, что многочисленные модули команды загружены в ядро. О том, как включить указанный режим, рассказывалось в разделах 28.4 и 12.12. Во всех пакетах, где есть команда `iptables`, имеются и сценарии загрузки ее модулей.

Нашим первым действием будет инициализация таблицы `filter`. Сначала все цепочки таблицы очищаются, после чего для цепочек `INPUT` и `OUTPUT` назначается стандартная директива `DROP`. Как и в других подобных случаях, наиболее безопасная стратегия заключается в удалении всех пакетов, которые не упомянуты явно.

```
iptables -F
iptables -P INPUT DROP
iptables -P FORWARD DROP
```

Поскольку правила проверяются по порядку, поместим наиболее часто используемые правила впереди[14]. Первые правила цепочки `FORWARD` разрешают устанавливать соединения с сетевыми службами по адресу 10.1.1.2. В частности, разрешаются протоколы SSH (порт 22), HTTP (порт 80) и HTTPS (порт 443). Самое первое правило пропускает все пакеты, исходящие из внутренней сети.

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 22 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p tcp --dport 443 -j ACCEPT
```

Единственный TCP-трафик, который может приниматься хостом (10.1.1.1), — это трафик протокола SSH, с помощью которого можно управлять брандмауэром. Второе из показанных ниже правил разрешает трафик интерфейса обратной связи внутри бранд-

[14] Следите за тем, чтобы стремление к повышению производительности не нарушало логику выполнения правил.

мауэра. Наши администраторы начинают нервничать, если им не удается с помощью команды `ping` найти стандартный маршрут, поэтому третье правило пропускает ICMP-пакеты `ECHO_REQUEST`, имеющие локальный IP-адрес отправителя.

```
iptables -A INPUT -i eth0 -d 10.1.1.1 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i lo -d 127.0.0.1 -p ANY -j ACCEPT
iptables -A INPUT -i eth0 -d 10.1.1.1 -p icmp --icmp-type 8 -j ACCEPT
```

Компьютеру, имеющему выход в Internet, для нормальной работы необходимо, чтобы брандмауэр пропускал определенные типы ICMP-пакетов. Следующие восемь правил пропускают минимальный набор ICMP-пакетов как на сам брандмауэр, так и в защищаемую им сеть.

```
iptables -A INPUT -p icmp --icmp-type 0 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 3 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 5 -j ACCEPT
iptables -A INPUT -p icmp --icmp-type 11 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 0 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 3 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 5 -j ACCEPT
iptables -A FORWARD -d 10.1.1.2 -p icmp --icmp-type 11 -j ACCEPT
```

Теперь добавим правила в цепочку `PREROUTING` таблицы `nat`. Эти правила не связаны с фильтрацией пакетов, зато полезны для защиты от поддельных IP-адресов. Правила с директивой `DROP` не нужно дублировать в цепочках `INPUT` и `FORWARDING`, поскольку цепочка `PREROUTING` применяется ко всем пакетам, поступающим на брандмауэр.

```
iptables -t nat -A PREROUTING -i ppp0 -s 10.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 172.16.0.0/12 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 192.168.0.0/16 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 127.0.0.0/8 -j DROP
iptables -t nat -A PREROUTING -i ppp0 -s 224.0.0.0/4 -j DROP
```

Наконец, завершим цепочки `INPUT` и `FORWARD` правилами, которые запрещают все пакеты, не упомянутые явно. Вообще-то, это уже было сделано с помощью команд `iptables -P`, но директива `LOG` заодно позволяет увидеть, кто к нам "стучится" из Internet.

```
iptables -A INPUT -i ppp0 -j LOG
iptables -A FORWARD -i ppp0 -j LOG
```

При желании можно включить систему NAT, чтобы скрывать частные адреса, используемые во внутренней сети. Подробнее об этом рассказывалось в разделе 12.12.

Одной из наиболее интересных возможностей механизма Netfilter является фильтрация пакетов с учетом состояния. Вместо того чтобы пропускать входящий трафик определенных типов, брандмауэр пропускает трафик, выдаваемый в ответ на клиентские запросы. Показанная ниже простейшая цепочка разрешает всем пакетам покидать сеть, но принимаются лишь те пакеты, которые поступают в рамках соединений, установленных хостами нашей сети.

```
iptables -A FORWARD -i eth0 -p ANY -j ACCEPT
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Чтобы команда `iptables` могла отслеживать сложные сетевые сеансы, например протоколов FTP и IRC, должны быть загружены определенные модули ядра. В противном случае команда просто не будет позволять устанавливать соответствующие соединения. Фильтрация пакетов с учетом состояния способна повысить безопасность системы,

но она резко повышает и сложность управления сетью. Убедитесь в том, что она действительно нужна, прежде чем реализовывать ее на брандмауэре.

Для отладки цепочек правил лучше всего подходит команда `iptables -L -v`. Она сообщит, сколько раз для того или иного правила было найдено совпадение. Мы часто добавляем временные правила с директивой `LOG`, если хотим получить дополнительную информацию о проверке пакетов. Более сложные проблемы решаются с помощью анализатора пакетов, такого как `tcpdump`.

20.14. Виртуальные частные сети

Одной из самых интересных разработок последних лет стало изобретение виртуальных частных сетей (Virtual Private Networks, VPN). Эта технология стала возможной в основном благодаря резкому возрастанию вычислительных мощностей процессоров (и рабочих станций пользователей). В своем простейшем виде виртуальная частная сеть (ВЧС) — это специальный тип сетевого соединения, эмулирующего непосредственное подключение удаленной сети, даже если в действительности она находится за тысячу километров и скрыта за множеством маршрутизаторов. Для повышения безопасности соединение не только подвергается аутентификации в том или ином виде (обычно с использованием совместного секретного ключа, например пароля), но еще и шифруется. Это называется *защищенным туннелем*.

Приведем пример ситуации, в которой ВЧС оказывается очень удобной. Предположим, компания имеет офисы в нескольких городах: Чикаго, Майами и нашем родном Боулдере. Если каждый из офисов подключен к локальному Internet-провайдеру, то посредством виртуальных частных сетей компания может прозрачным образом (и, главное, весьма безопасно) соединить все офисы через такую ненадежную сеть, как Internet. Конечно, такого же результата можно достичь и за счет покупки выделенных линий, но это обойдется гораздо дороже.

В качестве еще одного примера можно взять компанию, служащие которой выходят в сеть из дому. Наличие виртуальных частных сетей позволит служащим пользоваться преимуществами своих высокоскоростных и недорогих кабельных модемов и в то же время работать так, будто они непосредственно подключены к корпоративной сети.

Из-за удобства и популярности такого подхода многие компании стали предлагать решения, связанные с ВЧС. Функции поддержки ВЧС могут быть реализованы в маршрутизаторе, в операционной системе. Можно даже купить специализированное сетевое устройство, предназначенное специально для реализации ВЧС. Если позволяет бюджет, рассмотрите коммерческие предложения, в избытке имеющиеся на рынке.

Если же вы ограничены в средствах, обратитесь к пакету SSH, который позволяет организовывать защищенные туннели. Обычно он разрешает в каждый момент времени работать только с одним портом, но можно реализовать и псевдо-ВЧС, пример которой приводился в конце раздела 12.13. Там через SSH-туннель передавался трафик протокола PPP.

Туннели IPsec

Те, кого интересует надежное и недорогое решение в области ВЧС, должны обратить внимание на IPsec (Internet Protocol Security — механизм защиты протокола IP). Изначально он был реализован для протокола IPv6, но теперь доступен также и для IPv4. IPSec — это одобренная организацией IETF система аутентификации и шифрова-

ния соединений. Практически все серьезные поставщики ВЧС-систем оснащают свои продукты, по крайней мере, режимом совместимости с IPsec

Аутентификация позволяет убедиться в том, что пакет действительно пришел от заявленного отправителя и не был изменен в процессе передачи. Шифрование предотвращает несанкционированный просмотр содержимого пакета.

В своем нынешнем виде IPsec шифрует заголовок транспортного уровня, содержащий номера портов отправителя и получателя, что напрямую конфликтует со способом работы большинства брандмауэров. В настоящее время в организации IETF рассматривают вопрос об отмене этого свойства.

Ядра Linux версии 2.5.47 и выше включают свою собственную реализацию IPSec, которая полностью отличается от реализации FreeS/WAN, часто используемой с ядрами серии 2.4. Поскольку система IPsec является частью ядра, она поставляется со всеми рассматриваемыми нами дистрибутивами.

Существует одна тонкость, связанная с туннелями IPsec и параметром MTU передаваемых пакетов. Важно убедиться в том, что пакет, зашифрованный средствами IPsec, не подвергается фрагментации при прохождении туннеля. Для этого может потребоваться снизить значение MTU для сетевых интерфейсов, расположенных перед туннелем (обычно подходит значение 1400 байт). Подробнее о параметре MTU рассказывалось в разделе 12.3.

Так ли уж нужны виртуальные частные сети?

К сожалению, у виртуальных частных сетей есть и недостатки. Они позволяют организовать достаточно надежный туннель через ненадежный канал связи между двумя конечными точками, но они, как правило, не защищают сами конечные хосты. Например, если создать ВЧС между корпоративной магистралью и домашним компьютером президента компании, то можно ненароком открыть удобную лазейку для 15-летнего вундеркинда, по ночам наведывающегося в папин кабинет. Хорошо, если он использует эту возможность лишь для игр по сети с однокашниками. А если нет?

Отсюда вывод: соединения, устанавливаемые через туннели ВЧС, следует рассматривать как внешние и предоставлять им дополнительные привилегии лишь в случае крайней необходимости, тщательно взвесив все за и против. Можно добавить в свод правил безопасности, принятый в организации, специальный пункт, касающийся работы в ВЧС.

20.15. Специальные "защищенные" дистрибутивы Linux

К счастью (?), в последнее время делается все больше попыток выпустить "защищенные" версии Linux с увеличенным числом средств защиты по сравнению с обычными дистрибутивами. Усиление защиты достигается за счет использования специальных средств контроля доступа и других проверок. Очевидно, их полезнее всего применять в Linux-системах, внедренных в различные сетевые устройства. В то же время неясно, какие преимущества могут извлечь из этого рядовые пользователи. Такие системы все равно требуют тщательной настройки, наличия хорошего фильтра пакетов и многого другого, о чем шла речь в этой главе. Возможно, самая большая их польза заключается в повышении душевного спокойствия пользователей.

О SELinux более подробно рассказывалось в разделе 20.10.

В табл. 20.3 перечислен ряд наиболее известных "защищенных" дистрибутивов.

Таблица 20.3. "Защищенные" дистрибутивы Linux

Название проекта	Web-сайт
Bastille Linux	`www.bastille-linux.org`
Engarde Linux	`www.engardelinux.com`
Openwall GNU/*/Linux	`www.openwall.com/Owl`

20.16. Что нужно делать в случае атаки на сервер

Для начала — не паникуйте. Скорее всего, к моменту обнаружения взлома большая часть ущерба уже нанесена. Возможно, хакер "гулял" по системе несколько недель или даже месяцев. Вероятность того, что вам удалось выявить факт взлома, произошедший всего час назад, близка к нулю.

В свете этого мудрая сова приказывает сделать глубокий вздох и начать тщательно продумывать стратегию устранения взлома. Старайтесь не вспугнуть злоумышленника, во всеуслышание заявляя о происшествии или выполняя действия, которые могут насторожить того, кто наблюдал за деятельностью организации на протяжении нескольких недель. Подсказка: в этот момент неплохо выполнить резервное копирование. Надеемся, это не удивит нарушителя[15]?!

Стоит также напомнить самому себе о том, что, согласно исследованиям, в 60% инцидентов, связанных с нарушением конфиденциальности, присутствовал "внутренний враг". Не изучив всех фактов, старайтесь не посвящать в проблему лишних людей, кем бы они ни были.

Приведем короткий план, который поможет администратору пережить кризис.

Этап 1: перестаньте паниковать. Во многих случаях проблема обнаруживается только спустя какое-то время. Еще один-два часа или даже дня уже ничего не решат. Но имеет значение, какой будет реакция на событие: панической или рациональной. Часто в результате возникшей паники ситуация только усугубляется уничтожением важной информации, которая позволяет выявить нарушителя.

Этап 2: определите адекватную реакцию. Никому не выгодно раздувание инцидента, поэтому будьте благоразумны. Решите, кто из персонала будет участвовать в решении возникшей проблемы, и какие ресурсы при этом должны быть задействованы. Остальные будут помогать осуществлять "вынос тела", когда все закончится.

Этап 3: соберите всю возможную информацию. Проверьте учетные и журнальные файлы. Попытайтесь определить, где первоначально произошел взлом. Выполните резервное копирование всех систем. Убедитесь в том, что резервные ленты физически защищены от записи, если вы вставляете их в дисковод для чтения.

Этап 4: оцените нанесенный ущерб. Определите, какая важная информация (если таковая имеется) "покинула" компанию, и выработайте подходящую стратегию смягчения возможных последствий. Оцените степень будущего риска.

[15] Если создание резервных копий не является "нормальным" действием в вашей организации, то обнаружение взлома окажется самой незначительной из ожидающих вас проблем.

Этап 5: выдерните шнур. Если это необходимо и возможно, отключите "взломанные" компьютеры от сети. Перекройте обнаруженные "дыры". В документе CERT приведены инструкции относительно того, что следует делать в случае взлома системы. Найти этот документ можно по адресу `www.cert.org/tech_tips/win-UNIX-system_compromise.html`.

Этап 6: разработайте стратегию восстановления. Соберите толковых людей и обсудите план восстановления. Не заносите его в компьютер. Сосредоточьтесь на том, чтобы потушить "пожар" и свести ущерб к минимуму. Избегайте кого-либо обвинять или нагнетать обстановку. Не забудьте учесть психологический удар, который могут испытать пользователи. Пользователи имеют склонность доверять другим людям, и грубое нарушение доверия заставляет многих из них чувствовать себя неудобно.

Этап 7: сообщите о своем плане. Расскажите пользователям и управляющему персоналу о последствиях взлома, возможных будущих проблемах и предварительной стратегии восстановления. Будьте честны и откровенны. Инциденты, связанные с безопасностью, являются неотъемлемой частью современных сетей. Это не отражение ваших способностей как системного администратора или чего-то другого, чего можно было бы стыдиться. Открытое признание возникшей проблемы — 90% победы, если при этом вы демонстрируете, что у вас есть план выхода из ситуации.

Этап 8: воплотите план в жизнь. Вы знаете свои системы и сети лучше, чем кто бы то ни было. Доверьтесь плану и инстинктам. Посоветуйтесь с коллегами из других организаций, чтобы убедиться в правильности выбранного направления.

Этап 9: сообщите об инциденте в соответствующие органы. Если в инциденте участвовал "внешний игрок", сообщите об этом случае в организацию CERT. Связаться с ними можно либо по телефону горячей линии (412) 268-7090 и либо по электронной почте по адресу `cert@cert.org`. Предоставьте им как можно больше информации.

Стандартная форма для отчета находится на Web-сайте `www.cert.org`. Ниже приведен список того, что желательно включить в отчет:

- модели и платформы "взломанных" компьютеров, а также установленные на них операционные системы;
- список "заплат", которые были установлены в системе на момент инцидента;
- список учетных записей, подвергшихся нападению;
- имена и IP-адреса удаленных компьютеров, вовлеченных в инцидент;
- контактная информация, если известны администраторы удаленных систем;
- соответствующие журнальные записи и учетные данные.

Если есть подозрение, что вами обнаружена ранее неизвестная проблема в программном обеспечении, следует также сообщить об этом своему дистрибьютору Linux.

20.17. Источники информации, касающейся безопасности

В битве за безопасность системы половина успеха — знание современных разработок в этой области. Если система оказалась взломанной, это вряд ли стало следствием применения технологической новинки. Скорее всего, эта маленькая брешь в броне системы является известным уязвимым местом, и уже не раз описывалась в базе знаний произво-

дителя, в соответствующих посвященных вопросам безопасности группах новостей или списках рассылки.

Организация CERT

В связи с бурной реакцией общественности на вторжение Internet-червя в 1988 г. организация DARPA (Defense Advanced Research Projects Agency — Управление перспективных исследовательских программ Министерства обороны США) учредила новую структуру: CERT (Computer Emergency Response Team — группа компьютерной "скорой помощи"). В ее обязанности входит сбор информации, касающейся компьютерной безопасности. До сих пор это наиболее известный консультативный орган, хотя со временем он стал весьма медлительным и бюрократичным. Более того, в самой организации сейчас настаивают на том, что ее название не означает ничего кроме "зарегистрированной служебной марки университета Карнеги-Меллона".

В середине 2003 г. CERT подписала контракт о партнерстве слиянии с Национальным отделом по кибербезопасности Агентства внутренней безопасности (National Cyber Security Division, NCSD). Это слияние привело к изменению существовавшей ранее структуры списков рассылки. Теперь объединенная организация, известная как US-CERT, предлагает целых четыре списка объявлений, наиболее полезным из которых является список *Technical Cyber Security Alerts* (Сигналы о техническом нарушении кибербезопасности). Подписаться на эти списки можно по адресу `forms.us-cert.gov/maillists`.

Сервер SecurityFocus.com и список рассылки BugTraq

Сервер SecurityFocus.com специализируется на сборе новостей и различной информации по вопросам безопасности. В новостях публикуются как общие обзоры, так и статьи, посвященные конкретным проблемам. Есть также обширная библиотека полезных документов, удобно отсортированных по тематике.

В архиве сервера хранятся инструментальные средства для множества операционных систем. Программы снабжаются аннотациями и пользовательскими рейтингами. Это наиболее глобальный и исчерпывающий из известных нам архивов такого рода.

Список рассылки BugTraq — это форум для обсуждения проблем безопасности и путей их устранения. Чтобы подписаться, посетите страницу `www.securityfocus.com/archive`.

Объем информации, рассылаемый по списку, однако, может оказаться достаточно большим, а количество содержащихся в ней полезных сведений — довольно незначительным. На Web-сайте хранится база данных с отчетами о рассмотренных в форуме проблемах.

Информационный бюллетень Crypto-Gram

Ежемесячный бюллетень Crypto-Gram является ценным и временами даже забавным источником информации, касающейся компьютерной безопасности и криптографии. Его автором является Брюс Шнайер (Bruce Schneier), которые написал такие хорошо известные книги как *Applied Cryptography* (Криптография на практике) и *Secrets and Lies* (Секреты и мифы). Подписаться на бюллетень можно по следующему адресу:

```
www.schneier.com/crypto-gram.html
```

Посвященный вопросам безопасности блог Брюса доступен по адресу

`www.schneier.com/blog`

Организация SANS

SANS (System Administration, Networking, and Security Institute — институт системного и сетевого администрирования и проблем безопасности) — это профессиональная организация, которая спонсирует конференции и обучающие семинары, посвященные вопросам безопасности, а также публикует различную информацию по данной тематике. Ее Web-сайт `www.sans.org` является ценным информационным ресурсом, занимающим промежуточное положение между серверами SecurityFocus.com и CERT: не столь исчерпывающий, как первый, но и не такой скудный, как второй.

Организация SANS рассылает по электронной почте ряд еженедельных и ежемесячных бюллетеней, на которые можно подписаться на ее Web-сайте.

Информационные ресурсы отдельных дистрибутивов

Проблемы, касающиеся безопасности, очень влияют на репутацию, поэтому поставщики операционных систем остро заинтересованы в том, чтобы помочь пользователям сделать свои системы безопасными. Крупные поставщики ведут собственные списки рассылки и публикуют в них бюллетени по вопросам безопасности. Эта же информация часто доступна и на Web-сайтах. Не редкость, когда "заплаты" безопасности распространяются бесплатно, даже если поставщики обычно взимают плату за программную поддержку.

Есть Web-порталы, например `www.securityfocus.com`, на которых содержится информация, касающаяся конкретных производителей, и имеются ссылки на последние официальные пресс-релизы.

Список рекомендаций по вопросам защиты системы Red Hat можно найти по адресу `www.redhat.com/security`. На момент написания книги не существовал официальный список рассылки по данной тематике, спонсируемый поставщиком Red Hat. Но в Internet есть множество ресурсов, касающихся Linux. Большая часть информации в них напрямую применима в Red Hat.

Рекомендации по поводу защиты SuSE можно найти по адресу

`www.novell.com/linux/security/securitysupport.html`

Чтобы подписаться на официальный список рассылки, посвященный вопросам безопасности SuSE, посетите страницу

`www.suse.com/en/private/support/online_help/mailinglists/index.html`

Чтобы быть в курсе последних новостей, касающихся безопасности Debian, посетите Web-хост `www.debian.org` или подпишитесь на список рассылки по адресу

`www.debian.org/MailingLists/subscribe#debian-security-announce`

Список рассылки, посвященный безопасности Ubuntu доступен по адресу

`https://lists.ubuntu.com/mailman/listinfo/ubuntu-security-announce`

Информация о безопасности продуктов Cisco распространяется в виде практических замечаний, список которых находится по адресу `www.cisco.com/public/support/tac/fn_index.html` вместе с лентой агрегирования новостей. Чтобы подписаться на список рассылки, посвященный вопросам безопасности продуктов Cisco, направьте письмо по адресу `majordomo@cisco.com`, включив в тело сообщения строку "subscribe cust-security-announce".

Другие списки рассылки и Web-сайты

Перечисленные выше контактные адреса — лишь небольшая часть Internet-ресурсов по вопросам безопасности. Учитывая объем имеющейся сегодня информации, а также скорость, с которой появляются и исчезают различные ресурсы, мы посчитали целесообразным указать несколько "метаресурсов".

Хорошей стартовой площадкой является Web-хост X-Force (`xforce.iss.net`) компании Internet Security Systems, на котором содержится множество полезных списков часто задаваемых вопросов. Там же можно найти текущий перечень списков рассылки по вопросам безопасности, а также информацию по конкретным операционным системам и "заплатам" к ним.

И, конечно же, нельзя не упомянуть портал `www.yahoo.com`, содержащий множество полезных ссылок на интересующую нас тему. Лучше искать их в общем разделе "Security and Encryption" (Безопасность и шифрование) в Yahoo! Directory. Еще один хороший источник ссылок на тему сетевой безопасности можно найти на Web-сайте `www.wikipedia.org` под заголовком "Computer Security" (Компьютерная безопасность).

В журнале Linux Journal (`www.linuxjournal.com`) имеется прекрасная колонка под названием *Paranoid Penguin*, в которой рассматриваются все аспекты безопасности Linux. В ней также иногда публикуются различные специальные статьи, посвященные вопросам безопасности.

Сайт Linux Weekly News представляет собой "лакомый кусочек"; на нем всегда доступна свежая информация о защите ядра, дистрибутивах и других подобных вещах. Посвященный именно безопасности раздел этого сайта находится по адресу `lwn.net/security`.

20.18. Рекомендуемая литература

- Bryant, William. "Designing an Authentication System: a Dialogue in Four Scenes." 1988. `web.mit.edu/kerberos/www/dialogue.html`
- CERT Coordination Center. "Intruder Detection Checklist."1999. `www.cert.org/tech_tips/intruder_detection_checklist.html`
- CERT Coordination Center. "UNIX Configuration Guidelines." 1997. `www.cert.org/tech_tips/unix_configuration_fuidelines.html`
- Cheswick, William R., Steven M. Bellovin and Aviel D. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker (Second Edition)*.Reading, MA; Addison-Wesley. 2000.
- Curtin, Matt, Marcus Ranum and Paul Robinson. "Internet Firewalls: Frequently Asked Questions." 2004. `www.interhack.net/pubs/fwfaq`
- Farmer, Dan, and Wietse Venema. "Improving the Security of Your Site by Breaking Into it." 1993. `www.deter.com/unix/papers/improve_by_breakin.html`

- Farrow, Rik, and Richard Power. *Network Defense article series*. 1998-2004. `www.spirit.com/Network`
- Fraser, B., Editor. *RFC2196: Site Security Handbook*. 1997. `www.rfc-editor.org`
- Bauer Michael. *Linux Server Security. 2nd Edition*. Sebastopol, CA): O'Reilly Media, 2005.
- Garfinkel, Simson, Gene Spafford and Alan Schwartz. *Practical UNIX and Internet Security. Third Edition*. Sebastopol, CA: O'Reilly Media, 2003.
- Barrett, Daniel J., Richard E. Silverman and Robert G. Byrnes. *Linux Security Cookbook*. Sebastopol, CA: издательство O'Reilly Media, 2003.
- Kerby, Fred, et al. "SANS Intrusion Detection and Response FAQ." SANS. 2003. `www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm`
- Mann, Scott, and Ellen L. Mitchell. *Linux System Security: The Administrator's Guide to Open Source Security Tools (2nd Edition)*. Upper Saddle River, NJ: Prentice Hall PTR. 2002.
- Morris, Robert, and Ken Thompson. "Password Security: A Case History." Communications of the ACM, 22 (11): 594-597, November 1979. Перепечатано в *UNIX System Manager's Manual,* 4.3 Berkeley Software Distribution. University of California, Berkeley. April 1986.
- Pichnarczyk, Karyn, Steve Weeber, and Richard Feingold. "UNIX Incident Guide: How to Detect an Intrusion." Computer Incident Advisory Capability, U.S. Department of Energy. 1994. `www.ciac.org/cgi-bin/index/documents`
- Ritchie, Dennis M. "On the Security of UNIX." May 1975. Перепечатано в *UNIX System Manager's Manual,* 4.3 Berkeley Software Distribution. University of California, Berkeley. April 1986.
- Schneier, Bruce. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York, NY: Wiley, 1995.
- Кен Томпсон (Thompson, Ken). *Reflections on Trusting Trust* в журнале *ACM Turing Award Lectures: The First Twenty Years 1966-1985*. Reading, MA: ACM Press (Addison-Wesley), 1987.
- Sonnenreich, Wes, and Tom Yates. *Building Linux and OpenBSD Firewalls*. New York, NY: J.W. Wiley. 2000.

  Это замечательная книга: она компактна, легко читается, содержит хорошие примеры и не лишена чувства юмора. Огорчение вызывает лишь то, что ее авторы не рекомендуют применять утилиту `sudo` для получения прав суперпользователя, считая ее трудной для использования и потому не стоящей таких усилий. Мы категорически не согласны с этим!

20.19. Упражнения

20.1. Каковы преимущества SSH-аутентификации с использованием паролей Linux по сравнению с SSH-аутентификацией посредством идентификационного пароля и пары ключей. Если один из методов обеспечивает большую безопасность, обязательно ли переходить на него?

20.2. Утилита Samhain находит файлы, которые подверглись изменениям.

а) Что необходимо сделать для инсталляции и применения этой утилиты в вашей системе?

б) Против каких видов атак утилита оказывается наиболее эффективной?

в) С какими видами атак утилита не сможет справиться?

г) Если хакер проник в систему, то как он может обмануть утилиту?

д) Какой вывод можно сделать, если утилита сообщает об изменении файла `/bin/login`, но размер этого файла и время его модификации остались прежними? А что, если программа `sum` выдает одинаковые значения для старой и новой версий? Как на счет `md5sum`?

☆ 20.3. Туннели SSH часто являются единственным средством передачи трафика на удаленный компьютер, к которому у вас нет административного доступа. Прочтите `man`-страницу команды `ssh` и составьте команду, которая организует туннельную пересылку трафика с порта 113 локального компьютера на порт 113 хоста `mail.remotenetwork.org`. Точкой перенаправления туннеля тоже должен быть хост `mail.remotenetwork.org`.

☆ 20.4. Проанализируйте один из последних преданных огласке инцидентов, связанных с компьютерной безопасностью. Найдите соответствующие "заплаты" или способы защиты компьютеров вашей лаборатории. Укажите, какими источниками информации вы пользовались, и какие действия следует предпринять для защиты системы.

☆☆ 20.5. Предварительно получив разрешение от администратора, инсталлируйте утилиту John the Ripper, которая ищет учетные записи со "слабыми" паролями.

а) Модифицируйте исходный код этой утилиты, чтобы она отображала только регистрационные имена, но не пароли.

б) Примените эту утилиту к файлу паролей вашей лаборатории (потребуется получить доступ к файлу `/etc/shadow`) и посмотрите, сколько паролей она сможет взломать.

в) Задайте собственный пароль в виде слова из словаря и заставьте утилиту John the Ripper проверить только вашу запись файла `/etc/shadow`. Сколько времени ушло на взлом пароля?

г) Попробуйте разные варианты написания (с заглавной буквы, с цифрой в конце, однобуквенный пароль и т.д.) и оцените возможности утилиты.

☆☆ 20.6. Выберите два компьютера в лаборатории и назначьте один из них мишенью, а второй — зондом.

а) Инсталлируйте на зондирующем компьютере утилиты `nmap` и Nessus и попытайтесь атаковать компьютер-мишень. Как можно обнаружить факт атаки?

б) Создайте на компьютере-мишени брандмауэр с помощью команды `iptables` для защиты от этих атак. Можно ли теперь обнаружить факт атаки? Если да, то как? Если нет, то почему?

в) Какими дополнительными средствами защиты можно воспользоваться? (Необходим доступ с правами суперпользователя.)

★★ 20.7. Группа по компьютерной безопасности недавно обнаружила большую "брешь" во многих современных и старых версиях демона `sendmail`. Выясните, как лучше всего защититься от неприятностей.

★★ 20.8. Программы с установленным битом setuid часто являются необходимыми. А вот setuid-сценариев интерпретатора команд следует избегать. Почему?

★★ 20.9. С помощью `tcpdump` перехватите FTP-трафик как для активных, так и для пассивных сеансов FTP. Как необходимость поддержки анонимного FTP-сервера влияет на политику брандмауэра в организации. Что должны были бы разрешать правила брандмауэра. (Необходим доступ с правами суперпользователя.)

★★ 20.10. Что разрешают и запрещают показанные ниже правила, сообщаемые командой `iptables`? Что можно сделать для повышения безопасности такой системы? (Подсказка: цепочки `OUTPUT` и `FORWARD` можно расширить.)

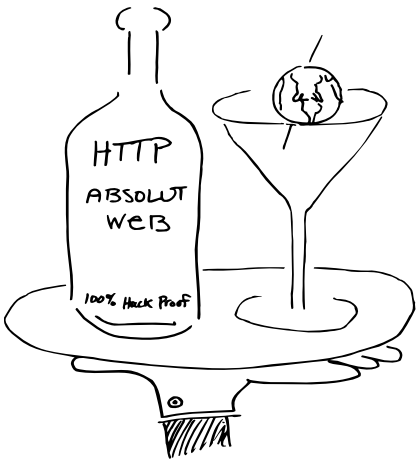
```
Chain INPUT (policy ACCEPT)
target  prot  opt  source   destination
block   all   --   anywhere  anywhere

Chain FORWARD (policy ACCEPT)
target  prot  opt  source   destination
       all   --   anywhere  anywhere

 Chain OUTPUT (policy ACCEPT)
target  prot  opt  source   destination

Chain block (1 references)
target  prot  opt  source       destination
ACCEPT  all   --   anywhere     anywhere   state RELATED,
ESTABLISHED
ACCEPT  tcp   --   anywhere     anywhere   state NEW tcp dpt:www
ACCEPT  tcp   --   anywhere     anywhere   state NEW tcp dpt:ssh
ACCEPT  tcp   --   128.138.0.0/16 anywhere   state NEW tcp dpt:
kerberos
ACCEPT  icmp  --   anywhere     anywhere
DROP    all   --   anywhere     anywhere
```

★★ 20.11. Проанализируйте цепочки правил вашего брандмауэра. Какие стратегии защиты они выражают? Все ли надежно в такой системе? (Выполнение этого упражнения может потребовать взаимодействия с администраторами, отвечающими за безопасность организации.)

★★★★★ 20.12. Напишите программу, которая находит сетевые интерфейсы, работающие в "беспорядочном" режиме. Регулярно запускайте эту программу для раннего обнаружения попыток вторжения в сеть. Какую нагрузку на сеть создает программа? Ее нужно запускать на каждом компьютере или достаточно одного? Можете ли вы составить такой пакет, который сообщит о том, что интерфейс находился в "беспорядочном" режиме? (Необходим доступ с правами суперпользователя.)

Глава 21

Web-хостинг и серверы Internet



Сложность Web-технологий, похоже, с каждым годом удваивается. К счастью, большинство связанных с ними задач находится в ведении Web-дизайнера или Web-программиста. Что касается Web-хостинга, то он за последние десять лет практически не изменился.

Дефекты, присутствовавшие в программном обеспечении Web-серверов, были почти все устранены, благодаря чему Web-серверы теперь являются довольно-таки безопасными и надежными, по крайней мере, если они правильно сконфигурированы и если в организации нет злонамеренных Web-программистов. Базовое программное обеспечение Web-серверов осталось почти таким же, даже несмотря на появление "Web 2.0", AJAX (Asynchronous JavaScript And XML — асинхронный JavaScript и XML) и динамического HTML.

В настоящее время на выбор доступны самые различные платформы Web-хостинга. Особенно усиленно среди них рекламируется платформа Windows производства Microsoft. В прессе постоянно публикуются статьи на тему "Какая платформа лучше всего подходит для Web-хостинга?", в которых, как правило, в роли главных конкурентов выступают Windows и Linux. И хотя некоторые из приводимых в этих статьях аргументов "за" или "против" выглядят просто как "Менее функциональна!" и "Классная система!", все-таки Linux признается самой популярной платформой для Web-хостинга из-за ее низкой стоимости, скорости, надежности и гибкости. Так называемая платфор-

ма LAMP (Linux, Apache, MySQL и PHP/Perl/Python) является доминирующей моделью для современных Web-служб.

Сегодня существует множество служб Internet, "хостинг" (т.е. размещение) которых может осуществляться либо на пользовательском сайте, либо на сервере одного из многочисленных провайдеров. В этой главе будут рассмотрены две самые главные службы: WWW и FTP.

21.1. Основы Web-хостинга

Хостинг Web-сайта мало чем отличается от других сетевых услуг. В основе WWW лежит HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) — простой протокол семейства TCP/IP, используемый для передачи документов, который содержит различные типы информации, включая текст, изображения, звук, анимацию и видео. HTTP аналогичен другим клиент-серверным протоколам сети Internet, таким как SMTP (электронная почта) и FTP (передача файлов).

Web-сервер — это система, сконфигурированная для ответа на HTTP-запросы. Чтобы преобразовать обычную Linux-систему в платформу Web-хостинга, необходимо запустить демон, который будет прослушивать TCP-порт с номером 80 (в соответствии со стандартом HTTP), принимать запросы на выдачу документов, а затем посылать эти документы пользователю.

Web-браузеры, такие как Firefox, Opera и Internet Explorer, подключаются к удаленным Web-серверам и делают запросы от имени пользователей. Полученные при этом документы могут содержать особые указатели (гиперссылки) на другие документы, находящиеся (или не находящиеся) на том сервере, к которому изначально подключился пользователь. Поскольку протокол HTTP полностью стандартизирован, клиенты, работающие в любой операционной системе на любой платформе, могут подключаться к любому серверу HTTP. Подобная независимость от платформы наряду с возможностью автоматического перенаправления пользователя от одного сервера к другому способствует повышению популярности протокола HTTP.

В настоящее время, однако, Web-хостинг "перешагнул" рамки протокола HTTP. Разработано много расширенных протоколов, обеспечивающих поддержку самых разных возможностей — от шифрования данных до потокового видео. Эти дополнительные службы зачастую контролируются разными демонами, даже если находятся на одном и том же физическом сервере.

Унифицированные указатели ресурсов

Унифицированный указатель ресурса (Uniform Resource Locator — URL) — это адрес объекта или службы в сети Internet. Он описывает порядок доступа к объекту при помощи пяти основных компонентов:

- протокола или приложения;
- имени хоста;
- порта TCP/IP (необязательно);
- каталога (необязательно);
- имени файла (необязательно).

В табл. 21.1 перечислены некоторые из протоколов, которые могут использоваться в URL-адресах.

Таблица 21.1. Протоколы, используемые в URL-адресах

Протокол	Назначение	Пример
file	Получение доступа к локальным файлам	file://etc/syslog.conf
ftp	Получение доступа к файлам по протоколу FTP	ftp://ftp.admin.com/adduser.tar.gz
http	Получение доступа к файлам по протоколу HTTP	http://admin.com/index.html
https	Получение доступа к файлам по протоколам HTTP/SSL	https://admin.com/order.shtml
ldap	Получение доступа к службе каталогов LDAP	ldap://ldap.bigfoot.com:389/cn=Herb
mailto	Отправка электронного сообщения по указанному адресу	mailto:linux@book.admin.com

Принцип работы HTTP

Как уже говорилось, в основе WWW лежит HTTP — удивительно простой клиент-серверный протокол, не хранящий информацию о состоянии сеанса. В модели HTTP инициатором подключения всегда является клиент (обычно браузер). Клиент запрашивает у сервера "содержимое" заданного URL-адреса. Сервер отвечает, передавая поток данных или возвращая сообщение об ошибке. После этого клиент может переходить к запросу еще одного объекта.

Поскольку протокол HTTP довольно прост, можно легко подключиться к Web-браузеру с помощью утилиты `telnet`. Для этого достаточно подключиться к порту 80 выбранного Web-сервера. После установления соединения сервер готов принимать HTTP-команды. Чаще всего используется команда `GET`, которая запрашивает содержимое документа. Обычно она задается в формате `GET /`. В этом случае возвращается корневой документ (как правило, начальная страница) сервера. Протокол HTTP является чувствительным к регистру символов, поэтому команды нужно вводить только прописными буквами.

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET /
<далее идет содержимое стандартного файла>
Connection closed by foreign host
```

Более "полный" HTTP-запрос включал бы номер версии HTTP-протокола, имя хоста, для которого предназначен запрос (необходимое для извлечения файла с виртуального хоста, основанного на именах) и другую информацию. А ответ тогда бы включал бы не только данные самого ответа, но и информационные заголовки. Например:

```
$ telnet localhost 80
Trying 127.0.0.1...
Connected to localhost.atrust.com.
Escape character is '^]'.
GET / HTTP/1.1
Host: www.atrust.com
```

```
HTTP/1.1 200 OK
Date: Sun, 06 Aug 2006 18:25:03 GMT
Server: Apache/1.3.33 (Unix) PHP/4.4.0
Last-Modified: Sun, 06 Aug 2006 18:24:49 GMT
Content-Length: 7044
Content-Type: text/html

<далее идет содержимое стандартного файла>
Connection closed by foreign host.
```

В данном случае мы сообщили серверу, что собираемся использовать протокол HTTP версии 1.1, и указали имя виртуального хоста, с которого хотим запросить информацию. Сервер вернул код состояния (`HTTP/1.1 200 OK`), текущую дату и время (в его представлении), имя и номер версии используемого им серверного программного обеспечения, дату последнего изменения запрашиваемого файла, его длину и тип содержащейся в нем информации. Данные заголовка отделены от содержимого файла одной пустой строкой.

Генерация содержимого на лету

Помимо обслуживания статических документов HTTP-сервер также может обеспечивать пользователя содержимым, генерируемым "на лету". Например, при желании сделать так, чтобы посещающим Web-сайт пользователям отображалось текущее время и температура воздуха, вы могли бы заставить HTTP-сервер выполнять специальный, извлекающий эту информацию сценарий. Этот замечательный "трюк" часто выполняется при помощи интерфейса CGI (Common Gateway Interface — общий шлюзовой интерфейс).

CGI представляет собой не язык программирования, а скорее спецификацию, в соответствии с которой HTTP-сервер обменивается информацией с другими программами. CGI-сценарии чаще всего пишутся на языке Perl, Python или PHP. Но в принципе для их написания подходит практически любой язык программирования, который может поддерживать операции ввода-вывода, выполняющиеся в режиме реального времени. Только представьте, скольким утратившим работу COBOL-программистам это позволит найти применение своим знаниям в Internet!

Помимо поддержки внешних CGI-сценариев многие Web-серверы также имеют такую архитектуру, которая позволяет встраивать прямо в них интерпретаторы сценариев типа Perl и PHP. В такой комплектации производительность Web-сервера значительно увеличивается, потому что у него отпадает необходимость порождать отдельный процесс для обработки каждого запроса сценария. Эта архитектура по большей части скрыта от глаз разработчиков сценариев. Каждый раз, когда сервер видит файл с указанным расширением (таким как `pl` или `php`), он отправляет содержимое этого файла встроенному интерпретатору для того, чтобы тот его выполнил.

CGI-сценарии и встраиваемые интерпретаторы являются главным образом заботой Web-разработчиков и программистов. Однако, к сожалению, они все-таки пересекаются с работой системных администраторов в одной такой важной области как безопасность. Поскольку CGI-сценарии и встраиваемые интерпретаторы имеют доступ к файлам, сетевым соединениям и прочим механизмам перемещения данных, их выполнение представляет потенциальную угрозу для безопасности компьютера, на котором установлен HTTP-сервер. Ведь все-таки они дают возможность любому человеку в мире выполнить на данном сервере какую-нибудь программу (или сценарий). Поэтому CGI-сценарии и файлы, обрабатываемые встроенными интерпретаторами, должны быть защищены не

хуже любой другой сетевой программы. Хорошим источником информации по обеспечению безопасности CGI-сценариев является страница `www.w3.org/Security/Faq`. И хотя эта страница некоторое время не обновлялась, вся выложенная на ней информация ничуть не утратила свою актуальность.

Балансировка нагрузки

Довольно трудно предсказать, сколько обращений (запросов объектов, изображений включительно) или операций просмотра страниц (запросов HTML-страниц) сервер может обработать за единицу времени. Все зависит от аппаратной платформы, установленной на ней операционной системы, ее настройки, а также, что, пожалуй, является наиболее важным, от организации Web-сайта (например, содержатся ли там статические HTML-страницы или выполняются запросы к базе данных и математические вычисления).

Ответить на вопрос о количестве обращений может позволить только непосредственное измерение производительности и уточнение фактических параметров сайта и применяемого аппаратного обеспечения. Иногда полезную для планирования информацию могут дать люди, работающие с аналогичными сайтами и на аналогичном оборудовании. Доверять цифрам, сообщаемым поставщиками систем, ни в коем случае не стоит. Также следует учитывать такой фактор, как пропускная способность. Один единственный компьютер, обрабатывающий HTML-файлы и изображения, запросто может обслуживать объем данных, достаточный для загрузки линии типа T3 (которая имеет пропускную способность 45 Мбайт/с).

Несмотря на все вышесказанное, лучше все-таки фокусироваться не на количестве обращений, а на степени масштабируемости: очень часто ресурсов ЦП и каналов ввода-вывода Web-серверу начинает не хватать гораздо раньше, чем ресурсов интерфейса Ethernet. Поэтому системным администраторам и Web-дизайнерам обязательно следует спланировать узел так, чтобы нагрузка распределялась между несколькими серверами.

Распределение нагрузки приводит к увеличению не только производительности, но и избыточности. Существует несколько способов распределения нагрузки, среди них: использование циклической службы DNS, использование специального распределяющего нагрузку оборудования и применение программных средств распределения нагрузки.

Использование циклической службы DNS является самым простым и самым примитивным способом балансировки нагрузки. В такой системе одному имени хоста назначается множество IP-адресов. Когда сервер имен получает запрос на IP-адрес Web-сайта, он передает клиенту в ответ один из этих IP-адресов. Адреса выдаются по очереди один за другим, в виде повторяющейся "циклической" последовательности.

О конфигурировании циклической DNS более подробно рассказывалось в конце раздела 15.5.

Проблемой подхода с использованием циклической службы DNS является то, что в случае сбоя в работе сервера данные DNS нужно будет обновить для того, чтобы удалить этот сервер из цикла ответа. Удаленное кэширование DNS-данных может сделать эту операцию сложной и ненадежной. При наличии резервного сервера, как правило, гораздо легче поступить следующим образом: переназначить IP-адрес отключившегося сервера резервному.

Использование аппаратных средств распределения нагрузки является довольно простым альтернативным вариантом, но требует наличия определенной дополнительной суммы денег. Среди коммерческих аппаратных средств распределения нагрузки наи-

большей популярностью пользуются: Big-IP Controller компании F5 Networks, Web-коммутаторы компании Nortlel и Content Services Switches компании Cisco. Эти продукты распределяют поступающие задачи в соответствии с рядом параметров и могут учитывать текущее время отклика отдельных серверов.

Программные средства распределения нагрузки не требуют наличия специализированного оборудования; они могут функционировать на сервере Linux.

Доступны как распространяемые с открытым исходным кодом, так и коммерческие решения. К числу средств с открытым исходным кодом относятся Linux Virtual Server (`www.linuxvirtualserver.org`), Ultra Monkey (`www.ultramonkey.org`) и модуль `mod_backhand` для Apache (`www.backhand.org`). Среди коммерческих средств наибольшей популярностью пользуются продукты, продаваемые компанией Zeus (`www.zeus.com`).

Вам наверняка интересно узнать, как же справляются с нагрузкой на таком большом Web-сайте, как Google. Они там используют специальные распределяющие нагрузку серверы DNS в сочетании с соответствующим распределяющим нагрузку оборудованием. Подробнее об этом можно узнать на сайте Wikipedia, в статье "Google platform" (Платформа Google).

Имейте в виду, что в настоящее время содержимое большинства сайтов генерируется динамически. Такая архитектура накладывает очень большую нагрузку на серверы баз данных. Если необходимо, проконсультируйтесь со своим администратором баз данных, чтобы определить наилучший способ распределения нагрузки между многочисленными серверами баз данных.

21.2. Инсталляция HTTP-сервера

В установке и сопровождении Web-сервера нет ничего сложного! Web-службы считаются гораздо менее сложными и неудобными в плане администрирования по сравнению со службами электронной почты и DNS.

Выбор сервера

В настоящее время доступно несколько видов HTTP-серверов, но, скорее всего, вы захотите начать с сервера Apache, который славиться своей гибкостью и производительностью. Он является стандартным Web-сервером в Red Hat, SuSE и Debian и отличается высокой гибкостью и производительностью. По состоянию на сентябрь 2006 г. 63% Web-серверов в Internet работали под управлением Apache. Остальные Web-серверы — а это приблизительно 30% — работали под управлением продуктов Microsoft. Такое распределение долей на рынке между Apache и Microsoft остается неизменным в течение уже пяти лет. Подробнее об изменениях рыночной доли различных Web-серверов можно узнать по адресу

```
news.netcraft.com/archives/web_server_survey.html
```

Полезные сравнительные данные существующих на текущий момент HTTP-серверов доступны по адресу `www.serverwatch.com/stypes/index.php` (в категории "Web Servers"). Ниже перечислены несколько факторов, которые не помешает учитывать при выборе:

- надежность;
- производительность;
- своевременность обновлений и исправлений ошибок;

- доступность исходного кода;
- уровень коммерческой или общественной поддержки;
- стоимость;
- безопасность и контроль доступа;
- возможность использования в качестве прокси-сервера;
- возможность шифрования данных.

Тем, кто умеет работать с исходным кодом, HTTP-сервер Apache обойдется бесплатно: весь его исходный код доступен на Web-сайте Apache Group по адресу `www.apache.org`. Те, кто побаивается иметь дело с "чистым" исходным кодом, возможно, захотят установить только бинарную версию пакета Apache, которая поставляется в виде части дистрибутива Linux. (Но не исключено, что она уже установлена; попробуйте поискать ее в каталоге `/etc/apache2`).

Установка сервера Apache

Если вы все-таки решили загрузить исходный код Apache и скомпилировать его самостоятельно, тогда сначала выполните сценарий `configure`, который входит в состав дистрибутива. Этот сценарий автоматически определяет тип операционной системы, а также устанавливает соответствующие `make`-файлы проекта. Чтобы указать, где именно в дереве каталогов должен размещаться сервер Apache, используйте опцию `--prefix`. Если вы этого не сделаете, сервер будет установлен в каталог `/usr/local/apache2`, который является каталогом по умолчанию. Ниже показан пример использования опции `--prefix`:

```
$ ./configure --prefix=/etc/httpd/
```

При помощи команды `configure--help` можно просмотреть список всех доступных опций. Большинство из этих опций будут представлять собой опции типа `--enable-`*`модуль`* и `--disable-`*`модуль`* и позволять включать и исключать различные функциональные компоненты, которые входят в состав Web-сервера.

Кроме того, можно еще также скомпилировать модули в файлы динамических совместно используемых объектов путем указания опции `--enable-`*`модуль`*`=shared` (или использования опции `--enabled-mods-shared=all`, если сделать совместно используемыми нужно все модули). Это позволит принимать решение о том, какие модули включить, а какие — исключить, позже: во время выполнения будут загружаться только те модули, которые указаны в конфигурации `httpd`. На самом деле в случае бинарной версии пакета Apache такая конфигурация даже является стандартной — абсолютно все модули компилируются в совместно используемые объекты и динамически загружаются при запуске сервера Apache. Недостатком такой конфигурации является только чуть более длительное время запуска и чуть более низкий показатель производительности (обычно не более чем на 5%). Для большинства сайтов возможность добавлять новые модули на лету и отключать существующие модули без повторной компиляции более чем оправдывает незначительное отставание в производительности.

Полный перечень стандартных модулей можно найти по адресу `httpd.apache.org/docs-2.0/mod`.

Хотя устанавливаемый по умолчанию набор модулей является вполне приемлемым, также может быть полезно активизировать и дополнительные модули, перечисленные в табл. 21.2.

Таблица 21.2. Полезные модули Apache, которые по умолчанию не включены

Модуль	Назначение
auth_dbm	Использует базу данных DBM для управления доступом пользователей/групп (рекомендуется, если требуется основанный на предоставлении каждым пользователем своего пароля доступ к различным частям Web-сайта)
rewrite	Переписывает URL-адреса при помощи регулярных выражений
expires	Позволяет включать в документ дату его истечения
proxy	Конфигурирует Apache в качестве прокси-сервера (подробнее об этой технологии рассказывается ниже)
ssl	Активизирует поддержку для протокола защищенных сокетов SSL (и протокола защищенной передачи гипертекста HTTPS)

Аналогичным образом может оказаться полезным отключить модули, перечисленные в табл. 21.3. Отключая модули, следует исходить из соображений безопасности и повышения производительности.

Таблица 21.3. Модули Apache, которые лучше удалить

Модуль	Назначение
asis	Позволяет посылать файлы указанных типов без использования HTTP-заголовков
autoindex	Отображает содержимое каталогов, в которых нет стандартного HTML-файла
env	Позволяет устанавливать специальные переменные среды для CGI-сценариев
include	Разрешает использовать включаемые файлы на стороне сервера, т.е. пользоваться схемой динамической генерации содержимого
userdir	Разрешает пользователям иметь собственные HTML-каталоги

Когда выполнение сценария `configure` завершится, выполните команды `make` и `make install`, чтобы фактически скомпилировать и установить соответствующие файлы.

Конфигурирование сервера Apache

Установив сервер, сконфигурируйте его в соответствии с требованиями имеющейся среды. Все конфигурационные файлы находятся в каталоге `conf` (например, `/usr/local/apache2/conf`). Проанализируйте и внесите необходимые изменения в файл `httpd.conf`, который состоит из трех разделов.

В первом разделе этого файла задаются глобальные настройки, такие как пул сервера, TCP-порт, через который HTTP-сервер должен принимать запросы (обычно это порт 80, но можно выбрать и другой порт, а также настроить на одном данном компьютере несколько HTTP-серверов и подключить их к разным портам), и параметры загрузки динамических модулей.

Во втором разделе конфигурируется сервер “по умолчанию”, т.е. сервер, который будет обрабатывать запросы, с которыми не сможет справиться ни один из виртуальных хостов `VirtualHost` (см. раздел 21.3). Здесь представлены такие конфигурационные параметры, как пользователь и группа, от имени которых будет запускаться сервер (главное, чтобы это был не пользователь root) и важнейший оператор `DocumentRoot`, который определяет корневой каталог обслуживаемых документов. В этом разделе также

задаются такие вещи, как обработка "специальных" URL-адресов, наподобие тех, что включают синтаксис *~имя_пользователя* для получения доступа к домашнему каталогу пользователя.

Глобальные параметры безопасности тоже устанавливаются во втором разделе конфигурационного файла. Директивы позволяют управлять доступом на уровне отдельных файлов (директива `<File>`) или на уровне отдельных каталогов (директива `<Directory>`). Именно с их помощью можно предотвратить доступ к важным файлам через `httpd`. Определить следует как минимум два уровня управления доступом: один — для файловой системы в общем и один — для главной папки документов. Предлагаемые в Apache параметры по умолчанию в принципе являются вполне приемлемыми, но мы все-таки рекомендуем удалить опцию `AllowSymLinks`, дабы предотвратить возможность использования демоном `httpd` в дереве документов символических ссылок. (Нам бы не хотелось, чтобы кто-то случайно создал символическую ссылку на каталог `/etc`, а вам?) Дополнительные рекомендации по обеспечению безопасности Apache можно найти по адресу:

```
httpd.apache.org/docs-2.0/misc/security_tips.html
```

В третьем и последнем разделе конфигурационного файла настраиваются виртуальные хосты. Об этом более подробно будет рассказываться в разделе 21.3.

После того, как внесете свои конфигурационные изменения, проверьте синтаксис конфигурационного файла, выполнив команду `httpd -t`. Если Apache отобразит сообщение "Syntax OK", тогда можете продолжать работать. В противном случае, внимательно проверьте файл `httpd.conf` на наличие опечаток и устраните их.

Запуск сервера Apache

Демон `httpd` может запускаться как вручную, так и при помощи системных сценариев запуска. Последний способ является более предпочтительным, поскольку он гарантирует, что Web-сервер будет перезапускаться всякий раз при перезагрузке компьютера. Чтобы запустить сервер вручную, следует ввести примерно такую команду:

```
$ /usr/local/apache2/bin/apachectl start
```

При желании сделать так, чтобы демон `httpd` запускался автоматически на этапе начальной загрузки, создайте в каталоге сценариев запуска системы (`rc`) символическую ссылку, указывающую на файл `/etc/init.d/httpd` (который устанавливается вместе с пакетом `httpd`). Лучше всего, чтобы демон `httpd` запускался после демонов, управляющих маршрутизацией и синхронизацией времени.

📖 О сценариях запуска системы рассказывалось в главе 2.

Анализ журнальных файлов

Внедрив Web-сайт в производственный процесс, вы наверняка пожелаете собирать статистические данные об использовании сайта, такие как количество запросов к каждой странице, среднее количество запросов за день, число неудачных запросов (в процентах) и количество переданных данных. Для этого вам сначала нужно будет сделать так, чтобы используемые журналы имели "комбинированный" формат (т.е., чтобы в конце директив `CustomLog` стояло слово `combined`, а не слово `common`). Такой формат подразумевает регистрацию в журналах информации об источнике каждого запроса

(страницы, с которой был вызван данный URL-адрес) и агенте пользователя (браузере и операционной системе клиента).

Журналы доступа и ошибок находятся на сервере Apache в каталоге `logs`. Эти файлы понятны для человека, но содержат так много информации, что на самом деле будет лучше установить специальную отдельную программу анализа для извлечения из них полезной информации. В настоящее время существуют буквально сотни различных программ-анализаторов журналов, как бесплатных, так и коммерческих.

Среди бесплатных анализаторов внимания заслуживают Analog (`www.analog.cx`) и AWStats (`awstats.sourceforge.ne`). Обе эти программы в принципе предоставляют только самые основные сведения. При желании иметь более детальные отчеты, пожалуй, придется приобретать какой-нибудь коммерческий пакет. Неплохой перечень программ анализа доступен по адресу `www.practicalapplications.net/kb/loganalysis.html`.

Оптимизация для обеспечения высокопроизводительного хостинга статического содержимого

За последние несколько лет специалисты по хостингу выяснили, что одним из самых простых способов создания высокопроизводительной платформы хостинга является оптимизация нескольких серверов для обслуживания статического содержимого. Для этого в Linux имеется уникальный механизм: Web-сервер TUX.

Этот сервер выполняется в адресном пространстве ядра и взаимодействует с традиционным Web-сервером, например Apache. Когда есть такая возможность, сервер TUX предоставляет статические страницы, не покидая адресного пространства ядра, как это делает демон `rpc.nfsd` при обслуживании файлов. Подобная организация работы устраняет необходимость копирования данных между ядром и программами пользовательского уровня, что сводит к минимуму число операций по переключению контекста. Сервер TUX — прекрасное средство для сайтов, от которых требуется молниеносное обслуживание статического содержимого.

Сервер TUX был выпущен для Red Hat (и доступен на Web-сайте `www.redhat.com`), но на условиях лицензии GPL, поэтому он может использоваться в других дистрибутивах Linux. Правда, настраивать его не так-то легко. Подробности можно узнать по адресу `http://www.redhat.com/docs/manuals/tux`.

21.3. Виртуальные интерфейсы

Раньше компьютер обычно служил сервером для одного Web-сайта (например, `www.acme.com`). По мере роста популярности WWW практически каждый пользователь обзавелся собственным Web-сайтом, и, как грибы после дождя, стали появляться тысячи новых компаний, занимающихся Web-хостингом.

Провайдеры быстро осознали, что можно добиться существенной экономии средств и ресурсов, если на одном сервере размещать несколько сайтов. Это позволит управлять группой сайтов, используя одно и то же аппаратное обеспечение. На практике такой подход реализуется с помощью *виртуальных интерфейсов*.

Виртуальные интерфейсы позволяют серверному демону идентифицировать соединения, основываясь не только на номере целевого порта (для протокола HTTP это порт 80), но и на целевом IP-адресе соединения. В настоящее время виртуальные интерфейсы используются достаточно широко и доказали свою эффективность не только в сфере Web-хостинга.

Идея, лежащая в основе виртуальных интерфейсов, весьма проста: одиночный компьютер обслуживает больше IP-адресов, чем позволяют его физические сетевые интерфейсы. Каждый из "виртуальных" сетевых интерфейсов может иметь доменное имя, под которым он известен пользователям Internet. Это позволяет единственному компьютеру обслуживать буквально сотни Web-сайтов.

Использование виртуальных хостов, основанных на именах

Протокол HTTP 1.1 тоже реализует функциональные возможности, подобные виртуальным интерфейсам (официально называемые "виртуальными хостами, основанными на именах"), которые устраняют потребность в назначении Web-серверам уникальных IP-адресов и конфигурировании специального интерфейса на уровне операционной системы. Этот подход позволяет совместно использовать IP-адреса, что особенно полезно, когда один сервер содержит сотни или тысячи начальных страниц (например, в случае университетских Web-сайтов).

К сожалению, такой подход нельзя назвать практичным для коммерческих сайтов, поскольку уменьшается степень их масштабируемости (приходится изменять IP-адрес при перемещении сайта на другой сервер) и возникает угроза для безопасности системы (если доступ к сайту фильтруется брандмауэром на основе IP-адресов). Кроме того, виртуальные хосты, основанные на именах, не могут использовать SSL. Так что, похоже, настоящие виртуальные интерфейсы будут применяться еще долго.

Конфигурирование виртуальных интерфейсов

Настройка виртуального интерфейса подразумевает два этапа. Сначала нужно создать виртуальный интерфейс на уровне TCP/IP. На втором этапе необходимо сообщить серверу Apache об имеющихся виртуальных интерфейсах. Этот второй этап более подробно описывается далее в этом разделе.

Виртуальные интерфейсы Linux обозначаются как *интерфейс*`:`*экземпляр*. Например, если интерфейс Ethernet называется `eth0`, то ассоциируемые с ним виртуальные интерфейсы могут называться так: `eth0:0`, `eth0:1` и т.д. Конфигурируются все интерфейсы с помощью команды `ifconfig`. Например, команда

```
# ifconfig eth0:0 128.138.243.150 netmask 255.255.255.192 up
```

конфигурирует интерфейс `eth0:0` и закрепляет за ним сетевой адрес 128.138.243.128/26.

В Red Hat и Fedora чтобы сделать назначенные виртуальные адреса постоянными, нужно для каждого виртуального интерфейса создать отдельный файл в каталоге `/etc/sysconfig/network-scripts`.

Например, файл `ifcfg-eth0:0`, соответствующий приведенной выше команде `ifconfig`, может содержать следующие строки:

```
DEVICE=eth0:0
IPADDR=128.138.243.150
NETMASK=255.255.255.192
NETWORK=128.138.243.128
BROADCAST=128.138.243.191
ONBOOT=yes
```

В Debian и Ubuntu применяется такой же подход, как и в Red Hat, но определения интерфейсов должны находиться в файле `/etc/network/interfaces`. В частности, записи для приведенного нами выше в качестве примера интерфейса `eth0:0` могут выглядеть так:

```
iface eth0:0 inet static
address 128.138.243.150
netmask 255.255.255.192
broadcast 128.138.243.191
```

В системах SUSE можно либо создавать виртуальные интерфейсы при помощи YaST, либо вручную создавать файлы интерфейсов.

В SUSE все IP-адреса интерфейса конфигурируются внутри одного единственного файла. Чтобы сконфигурировать файлы вручную, отыщите в каталоге `/etc/sysconfig/network` файлы, имена которых начинаются с комбинации `ifcfg-`*`ifимя`*. Имена файлов настоящих интерфейсов будут содержать громоздкие 6-байтовые MAC-адреса; именно они вам и нужны.

Например, один из конфигурационных файлов может содержать следующие записи для определения двух виртуальных интерфейсов:

```
IPADDR_0=128.138.243.149
NETMASK_0=255.255.255.192
LABEL_0=0
IPADDR_1=128.138.243.150
NETMASK_1=255.255.255.192
LABEL_1=1
STARTMODE="onboot"
NETWORK=128.138.243.128
```

Суффиксы после `IPADDR` и `NETMASK` (которые здесь выглядят как `_0` и `_1`) вовсе не обязательно должны быть числовыми, но по логическим соображениям такой формат считается удобным.

Уведомление Apache о наличии виртуальных интерфейсов

Создав виртуальные интерфейсы, нужно указать серверу Apache, какие документы он должен предоставлять при подключении клиента к каждому из этих интерфейсов (IP-адресов). Сделать это можно в файле `httpd.conf` при помощи конструкции `VirtualHost`. Для каждого сконфигурированного виртуального интерфейса следует добавить одну такую конструкцию. Например:

```
<VirtualHost 128.138.243.150>
  ServerName www.company.com
  ServerAdmin webmaster@www.company.com
  DocumentRoot /var/www/htdocs/company
  ErrorLog logs/www.company.com-error_log
  CustomLog logs/www.company.com-access_log combined
  ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

В этом примере любому клиенту, который подключится к виртуальному хосту 128.138.243.150, будут предоставляться документы из каталога `/var/www/htdocs/`

company. Для определения специфических параметров виртуального хоста в конструкции VirtualHost может использоваться почти любая директива Apache. Относительные пути к каталогам, включая те, что указаны для директив DocumentRoot, ErrorLog и CustomLog интерпретируются в контексте ServerRoot.

В случае виртуальных хостов, основанных на именах, все многочисленные DNS-имена указывают на один и тот же IP-адрес. Конфигурирование сервера Apache выполняется похоже, но подразумевает указание первичного IP-адреса, по которому Apache должен принимать поступающие запросы, и пропуск IP-адреса в конструкции VirtualHost:

```
NameVirtualHost 128.138.243.150
<VirtualHost *>
  ServerName www.company.com
  ServerAdmin webmaster@www.company.com
  DocumentRoot /var/www/htdocs/company
  ErrorLog logs/www.company.com-error_log
  CustomLog logs/www.company.com-access_log combined
  ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
</VirtualHost>
```

В этой конфигурации Apache проверяет HTTP-заголовки для определения запрашиваемого сайта. Сервер принимает запросы для www.company.com через свой главный IP-адрес 128.138.243.150.

21.4. Протокол защищенных сокетов (SSL)

Протокол SSL[1] защищает соединения между Web-сайтом и браузером клиента. В URL-адресах, начинающихся с https://, используется именно эта технология. Для защиты от перехвата, фальсификации и подделки данных SSL применяет шифрование.

Для установки соединения браузер и сервер используют основанную на сертификатах схему аутентификации, после чего переключаются на более быструю основанную на шифрах криптографическую схему для защиты фактически передаваемых через это соединение данных.

SSL запускается в виде отдельного уровня под протоколом HTTP. SSL просто обеспечивает безопасность для соединения и никоим образом не вмешивается в транзакцию HTTP. Благодаря такой своей "чистой" архитектуре SSL может применяться для защиты не только HTTP, но и таких протоколов, как SMTP, NNTP и FTP. Подробнее об этом можно узнать на сайте Wikipedia, указав в строке поиска слова "Secure Sockets Layer".

В "первые дни" использования SSL большинство ключей симметричного шифрования были довольно слабыми (40-битными) из-за установленных правительством США ограничений на экспорт криптографических технологий. Спустя годы, после многочисленных споров и судебных разбирательств, правительство смягчило некоторые из своих ограничений на экспорт, разрешив использовать в SSL-реализациях 128-битные ключи для симметричных шифров.

📖 Правовые вопросы, касающиеся шифрования, будут более подробно описываться в разделе 30.12.

[1] Протокол защиты транспортного уровня (Transport Layer Security, TLS) является следующей версией протокола SSL и реализован во всех современных браузерах. Однако Web-пользователи все равно для описания концепции в основном используют термин "SSL".

Генерация запроса на подпись сертификата

Владелец Web-сайта, который будет использовать SSL, обязательно должен сгенерировать специальный запрос на подпись сертификата (Certificate Signing Request, CRS), т.е. цифровой файл, содержащий криптографический ключ и название компании. Затем он должен отправить этот "сертификат" в центр сертификации (Certificate Authority, CA) "на подпись". После возвращения из центра сертификации сертификат помимо криптографического ключа сайта и названия компании должен также содержать одобрение этого центра сертификации.

У Web-браузеров имеются встроенные списки центров сертификации, сертификаты которых они могут принимать. Если ваш центр сертификации присутствует в списке центров сертификации браузера, браузер сможет проверить подпись на вашем сертификате и получить ваш криптографический ключ, а значит он сможет и отправлять сообщения, расшифровать которые сможет только ваш сайт. Хотя вы можете сами подписать свой сертификат, большинство браузеров при получении сертификата, подписанного нераспознанным центром сертификации, уведомляют пользователя о том, что данный сертификат является потенциально небезопасным. В мире коммерции такое поведение явно будет представлять проблему. Но если вы хотите создать свой собственный центр сертификации для внутреннего пользования и тестирования, советуем заглянуть на следующую страницу:

```
httpd.apache.org/docs/2.0/ssl/ssl_faq.html#aboutcerts.
```

Подписать сертификат можно в любом центре сертификации, которых сегодня существует довольно много. Просто введите в поле для поиска на сайте Google ключевые слова "SSL-сертификат" и затем выберите в выданных результатах любую понравившуюся компанию. Центры сертификации отличаются только объемом работы, которую они проделывают, чтобы проверить ваши идентификационные данные, гарантиями, которые они предлагают, и количеством браузеров, которые поддерживают их сразу же после установки (большинство центров сертификации поддерживается огромным числом различных браузеров).

Процедура создания сертификата для отправки в центр сертификации является довольно простой. Главное, чтобы был установлен пакет OpenSSL (который в большинстве дистрибутивов устанавливается по умолчанию). Вот что нужно сделать.

Во-первых, создайте для своего сервера Apache 1024-битный секретный ключ RSA:

```
$ openssl genrsa -des3 -out server.key 1024
```

Последует приглашение ввести и подтвердить идентификационную фразу для зашифровки серверного ключа. Сделайте резервную копию файла `server.key`, сохранив ее в каком-нибудь безопасном месте (доступном только для пользователя root), и не забудьте запомнить введенную идентификационную фразу. Любопытные могут просмотреть числовые детали ключа при помощи такой команды:

```
$ openssl rsa -noout -text -in server.key
```

Далее, создайте запрос CSR (Certificate Signing Request), включив в него сгенерированный только что серверный ключ:

```
$ openssl req -new -key server.key -out server.csr
```

Введите полностью уточненное доменное имя сервера, когда появится приглашение ввести "общее имя". Например, если URL-адрес сайта выглядит так: `https://www.`

company.com, в качестве общего имени следует ввести "www.company.com". Обратите внимание на то, что для каждого имени хоста требуется отдельный сертификат, даже если отличия между этими именами такие, как между "company.com" и "www.company.com". Компании обычно регистрируют только какое-нибудь одно общее имя и затем просто проверяют, чтобы все основанные на SSL ссылки указывали на него.

Просмотреть детали сгенерированного CSR-запроса можно при помощи следующей команды:

```
$ openssl req -noout -text -in server.csr
```

Теперь файл server.csr можно отправлять в выбранный центр сертификации на подпись. Сохранять локальную копию этого файла не обязательно. Подписанный CSR-файл, который вернет центр сертификации, должен иметь расширение .crt. Далее подписанный сертификат останется только поместить в то же безопасное место, в котором был сохранен секретный ключ.

Конфигурирование сервера Apache так, чтобы он использовал SSL

HTTP-запросы поступают через порт 80, а для HTTPS-запросов используется порт 443. И те, и другие могут обслуживаться одним и тем же процессом Apache. Однако SSL не работает с виртуальными хостами, основанными на именах; у каждого виртуального хоста должен быть конкретный IP-адрес. (Это ограничение является следствием дизайна SSL.)

Чтобы настроить Apache на использование SSL, сначала нужно проверить, активизирован ли модуль SSL в файле httpd.conf, отыскав и добавив следующую строку, если она отсутствует:

```
LoadModule ssl_module libexec/mod_ssl.so
```

Затем следует добавить директиву VirtualHost для порта SSL:

```
<VirtualHost 128.138.243.150:443>
  ServerName www.company.com
  ServerAdmin webmaster@www.company.com
  DocumentRoot /var/www/htdocs/company
  ErrorLog logs/www.company.com-ssl-error_log
  CustomLog logs/www.company.com-ssl-access_log combined
  ScriptAlias /cgi-bin/ /var/www/cgi-bin/company
  SSLEngine on
  SSLCertificateFile /usr/local/apache2/conf/ssl.crt/server.crt
  SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/server.key
</VirtualHost>
```

Обратите внимание на :443 после IP-адреса и SSL-директивы, указывающие Apache, где следует искать секретный ключ и подписанный сертификат.

После перезапуска сервера Apache, появится приглашение ввести идентификационную фразу для доступа к файлу server.key. Из-за этого приглашения httpd не сможет больше запускаться автоматически при загрузке компьютера. При желании избавиться от этого приглашения можно удалить шифрование из своего секретного ключа:

```
$ cp server.key server.key.orig
$ openssl rsa -in server.key.orig -out server.key
$ chmod 400 server.key server.key.orig
```

Конечно, любой, кому удастся получить копию вашего незашифрованного ключа, сможет затем выдавать ваш сайт за свой.

Подробнее об SSL можно узнать по следующим адресам:

```
httpd.apache.org/docs-2.0/ssl/ssl_faq.html
httpd.apache.org/docs/2.0/mod/mod_ssl.html
```

21.5. Кэширование и прокси-серверы

Рост сети Internet и объема находящейся в ней информации происходит по экспоненциальному закону. Следовательно, пропускная способность глобальных каналов и мощность вычислительных ресурсов, требуемых для их обслуживания, также возрастают согласно этому закону. Как укротить такую "стихию"?

Единственным методом обуздания подобного роста является использование репликации. Независимо от того, на каком уровне — национальном, региональном или корпоративном — находится информация, по мере роста сети Internet требуется перемещать данные на ближние узлы. Не имеет смысла передавать одну и ту же популярную Web-страницу, скажем, из Австралии в США миллионы раз в день, используя весьма дорогостоящий международный канал связи. Должен существовать способ сохранять эту информацию после ее однократной передачи по каналу. И такой способ есть.

Кэш Squid и прокси-сервер

Одним из возможных решений является распространяемый бесплатно пакет под названием "Squid" (которое в переводе означает "кальмар")[2]. Этот пакет представляет собой кэширующий сервер и прокси-сервер одновременно и поддерживает несколько протоколов, включая HTTP, FTP и SSL.

Схема работы пакета такова. Клиентские Web-браузеры подключаются к серверу Squid, чтобы запросить объект из Internet. Сервер выполняет запрос от имени клиента (либо извлекает объект из кэша) и возвращает результат клиенту. Прокси-серверы этого типа часто используются для усиления безопасности или фильтрации содержимого.

В системе, где работает прокси-сервер, непосредственный доступ в Internet через брандмауэр нужен лишь одному компьютеру. В компьютерных классах средних школ прокси-серверы зачастую выполняют фильтрацию содержимого, чтобы дети не могли получить доступ к неподобающим материалам. В настоящее время имеется множество коммерческих и бесплатных прокси-серверов (некоторые из них основаны на пакете Squid, а некоторые — нет). Одни из этих систем являются чисто программными (как Squid), другие поставляются в виде аппаратного устройства (как, например, BlueCoat; подробности — на сайте `www.cacheflow.com`). Обширный перечень существующих технологий прокси-сервера доступен по адресу:

```
www.web-caching.com/proxy-caches.html
```

Служба фильтрации — это, конечно, хорошо, но настоящим достоинством пакета Squid являются его кэширующие возможности. Сервер Squid не просто кэширует информацию, получаемую в результате выполнения запросов локальных пользователей, но

[2] Почему "кальмар"? По словам разработчиков, "все хорошие названия уже разобрали".

также позволяет формировать иерархию серверов Squid[3]. Группы серверов Squid применяют протокол ICP (Internet Cache Protocol — протокол кэширования в сети Internet) для передачи друг другу сведений о содержимом своих кэш-буферов.

Это позволяет администратору создать систему, в которой локальные пользователи взаимодействуют с кэширующим сервером организации с целью получения Web-содержимого. Если другой пользователь организации уже запрашивал эти же данные, то при повторном обращении к ним будет возвращена их копия, причем скорость передачи окажется сравнимой со скоростью работы локальной сети (обычно 100 Мбит/с или выше). При отсутствии на локальном сервере Squid требуемой информации она запрашивается у регионального кэширующего сервера. Как и в случае локального сервера, если какой-либо пользователь регионального сервера уже запрашивал объект раньше, то этот объект будет немедленно взят из кэша. При отсутствии объекта сервер обратится к вышестоящему серверу (обслуживающему страну или континент) и т.д. В результате производительность обработки запросов существенно повышается, и пользователи это чувствуют.

Применение пакета Squid экономически выгодно. Поскольку пользователи часто запрашивают в Internet одну и ту же информацию, в крупных организациях наблюдается значительное дублирование внешних Web-запросов. Эксперименты показали, что установка кэширующего сервера приводит к уменьшению внешнего трафика на 40%.

Поэтому, чтобы эффективно использовать Squid, скорее всего, вы заставите своих пользователей иметь дело с кэшем. Для этого вам нужно будет либо сконфигурировать прокси-сервер по умолчанию через Active Directory (если речь идет о среде Windows), либо настроить маршрутизатор для перенаправления всего Web-трафика в кэш Squid путем использования протокола WCCP (Web Cache Communication Protocol — протокол связи с Web-кэшем).

Инсталляция сервера Squid

Сервер Squid довольно легко устанавливается и конфигурируется. Поскольку ему необходимо свободное пространство для кэша, его следует устанавливать на отдельном компьютере с достаточным объемом оперативной и дисковой памяти. Конфигурация для относительно большого кэша предусматривает 2 Гбайт ОЗУ и 200 Гбайт пространства на жестком диске.

Пакет доступен в формате RPM или в формате утилиты `apt-get` для различных дистрибутивов Linux. Кроме того, свежую копию пакета Squid можно загрузить из сайта `www.squid-cache.org`. Во втором случае после распаковки инсталляционного дистрибутива запустите сценарий `configure`, находящийся в начальном каталоге. По умолчанию предполагается, что пакет будет установлен в каталог `/usr/local/squid`. Если требуется задать другой каталог, воспользуйтесь опцией `--prefix=`*`каталог`* сценария `configure`. После завершения сценария выполните команды `make` и `make install`.

После инсталляции Squid нужно локализировать конфигурационный файл `squid.conf`. Список изменений, которые следует внести в стандартный (т.е. поставляемый в качестве образца) файл `squid.conf`, можно найти в каталоге дистрибутива, в файле `QUICKSTART`.

[3] К сожалению, некоторые узлы помечают все свои страницы как некэшируемые, что препятствует эффективной работе сервера Squid. Кроме того, сервер не кэширует динамически формируемые страницы.

Также нужно выполнить вручную команду `squid -z`, чтобы сформировать и предварительно очистить структуру каталогов, в которой будут храниться кэшированные Web-страницы. В конце можно запустить сервер вручную с помощью сценария `RunCache`, хотя обычно он вызывается из сценариев запуска системы, чтобы сервер Squid автоматически запускался на этапе начальной загрузки.

Чтобы протестировать сервер Squid, придется сконфигурировать Web-браузер так, чтобы в качестве прокси-сервера он использовал сервер Squid. Позволяющая это сделать опция обычно находится в панели настроек браузера.

21.6. Установка анонимного FTP-сервера

FTP — один из старейших протоколов сети Internet, широко используемый и по сей день. Несмотря на то что протокол может применяться для решения различных внутренних задач организации, в Internet он чаще всего все-таки используется в виде так называемого "анонимного" FTP-сервера, с помощью которого пользователи могут загружать предоставленные им файлы, не имея учетной записи на данном узле.

FTP очень удобен для распространения программного обеспечения, документации, файлов "заплат" и т.д., но в наши дни почти все FTP-серверы заменили серверы HTTP. Аргументы в пользу FTP являются довольно слабыми: FTP-сервер может быть чуть более надежным, и для получения доступа к FTP-сайту пользователям не нужен Web-браузер (хотя конечно, в этом случае им необходим FTP-клиент).

Простой протокол FTP следует использовать *только* тогда, когда требуется анонимный доступ. Для не анонимных приложений следует использовать его безопасную версию — SFTP. FTP передает пароли в виде обычного текста и не раз приводил к инцидентам с безопасностью.

Демон `ftpd` управляется демоном `inetd` и потому имеет соответствующие записи в файлах `/etc/inetd.conf` и `/etc/services` (если вместо демона `inetd` используется демон `xinetd`, тогда запись должна существовать в файле `/etc/xinetd.d`). Когда пользователь FTP-сервера регистрируется анонимно, демон `ftpd` выполняет системный вызов `chroot` (сокращение от "change root"), чтобы сделать файлы, находящиеся за пределами каталога `~ftp`, невидимыми и недоступными. Из-за общедоступной природы анонимного FTP-сервера очень важно, чтобы демон `ftpd` был сконфигурирован правильно, так чтобы секретные файлы не стали случайно доступными всему миру.

Чтобы активизировать анонимный FTP-сервер на сайте, выполните следующие действия (четко следуя указанному порядку):

- добавьте запись псевдопользователя `ftp` в обычный файл паролей и в файл теневых паролей (в принципе во всех дистрибутивах, кроме Debian, пользователь `ftp` должен уже существовать). Поскольку никто не должен регистрироваться в системе под именем `ftp`, задайте для данной учетной записи в качестве пароля символ "x". Также будет неплохо, если в качестве стандартной оболочки для данного пользователя вы укажете `/sbin/nologin` или `/bin/false`;
- создайте домашний каталог для пользователя `ftp`, если таковой еще не существует;
- создайте в каталоге `~ftp` подкаталоги `bin`, `etc`, `lib` и `pub`. В сеансе доступа к анонимному FTP-серверу корневым каталогом системы становится `~ftp` (благодаря системному вызову `chroot`), поэтому подкаталоги `bin` и `etc` должны содержать копии всех утилит и конфигурационных файлов, необходимых демону `ftpd`.

После выполнения системного вызова chroot каталоги ~ftp/bin и ~ftp/etc выглядят как /bin и /etc;

- скопируйте программу /bin/ls в каталог ~ftp/bin. Для дополнительного повышения безопасности сделайте файл ~ftp/bin/ls доступным только для выполнения, назначив ему режим доступа 111. Это не позволит клиентам копировать двоичный код программы ls, чтобы попытаться найти в нем слабые места;
- скопируйте в каталог ~ftp/lib совместно используемые библиотеки, необходимые программе ls (или создайте жесткие ссылки на них). Посмотрите в документации к дистрибутиву, какие именно файлы требуются. Обратите внимание на то, что жесткие ссылки применимы только в том случае, когда файлы находятся в одном и том же разделе диска;
- скопируйте файлы /etc/passwd и /etc/group в каталог ~ftp/etc;
- отредактируйте файлы passwd и group. Демон ftpd пользуется только командой ls и усеченными копиями файлов /etc/passwd и /etc/group из каталога ~ftp/etc. Файлы passwd и group в каталоге ~ftp должны содержать только информацию о корневом каталоге, демоне и ftp, а также символ "x" в полях паролей;
- задайте надлежащие права доступа к файлам и подкаталогам в каталоге ~ftp. Мы рекомендуем установить их так, как показано в табл. 21.4;

Таблица 21.4. Рекомендуемые права доступа к содержимому каталога ~ftp

Файл/каталог	Владелец	Режим
~ftp	root	555
~ftp/bin	root	555
~ftp/bin/ls	root	111
~ftp/etc	root	555
~ftp/etc/passwd	root	444
~ftp/etc/group	root	444
~ftp/pub	root	755
~ftp/lib	root	555

- отредактируйте файл /etc/ftpusers, удалив из него записи ftp и anonymous для того, чтобы разрешить регистрироваться анонимным пользователям;
- поместите файлы, которые хотите сделать доступными, в каталог ~ftp/pub.

Одна из основных проблем, связанных с анонимным доступом по протоколу FTP, заключается в том, что пользователям разрешается хранить файлы в FTP-каталогах. Каталоги, доступные всему миру, каким бы благородным целям они не служили, становятся приманкой для хакеров и пиратов, распространяющих нелегальное программное обеспечение. Эти "деятели" захватывают всю полосу пропускания и втягивают организацию в деятельность, которая является как минимум нежелательной, а часто — просто незаконной. Будьте бдительны! Не допускайте наличия в системе FTP-каталогов, доступных для записи в анонимном режиме.

21.7. Упражнения

☆ 21.1. Сконфигурируйте виртуальный интерфейс на рабочей станции. Выполните команду `ifconfig` до и после этого и сравните результат. Можно ли обнаружить виртуальный интерфейс с помощью команды `ping`, выполненной на другом компьютере той же подсети? А другой сети? Объясните свой ответ. (Необходим доступ с правами суперпользователя.)

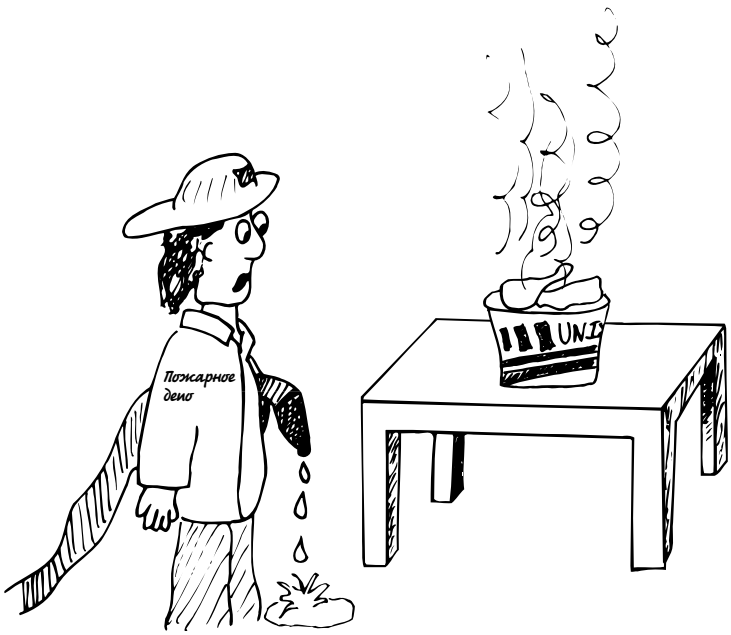
☆ 21.2. С помощью анализатора пакетов (`tcpdump`) прослушайте трафик двустороннего HTTP-диалога, связанного с передачей информации на сервер (например, с заполнением полей формы или поля поиска). Покажите, каким образом браузер посылает данные Web-серверу. (Необходим доступ с правами суперпользователя.)

☆ 21.3. Используйте анализатор пакетов для перехвата трафика, возникающего при открытии часто посещаемой Web-страницы, например начальной страницы портала `www.amazon.com` или `www.cnn.com`. Сколько отдельных TCP-соединений приходится открывать? Кто инициирует их? Является ли это наиболее эффективным способом использования системы? (Необходим доступ с правами суперпользователя.)

☆ 21.4. Найдите журнальные файлы Web-сервера и проанализируйте их. Запросы каких типов поступали серверу за последние несколько часов? Какие ошибки произошли за этот период? Что можно сказать о безопасности сервера на основании содержимого журнальных файлов? (Может потребоваться доступ с правами суперпользователя.)

☆☆ 21.5. Инсталлируйте сервер Apache и создайте несколько Web-страниц. Проверьте с других компьютеров, работает ли сервер. Найдите журнальные файлы Apache, которые позволяют узнать, какие браузеры обращались к серверу. Настройте сервер Apache на выдачу некоторых Web-страниц через виртуальный интерфейс, созданный в упражнении 21.1. (Необходим доступ с правами суперпользователя.)

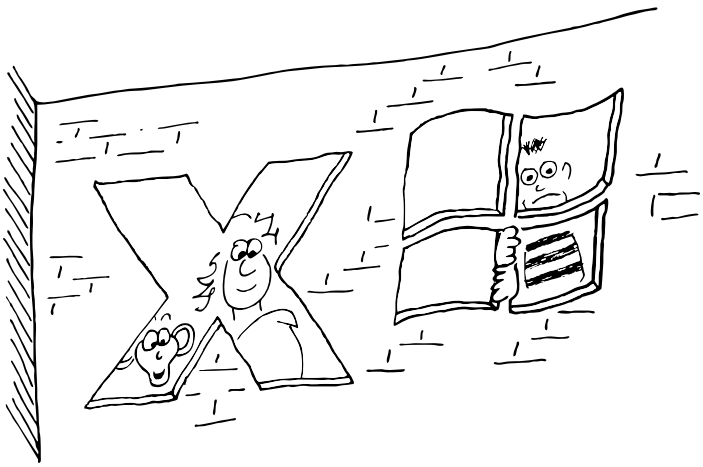
ЧАСТЬ III
Разное



Book_LinuxAdmHand.indb 814 17.04.2007 13:56:18

Глава 22

Система X Window System



Система X Window System (ее еще называют X11, или просто X) является основой для большинства графических пользовательских сред для систем UNIX и Linux. X — это естественная наследница оконной системы, называемой W (это действительно так), которая разрабатывалась под эгидой проекта Project Athena в Массачусетском технологическом институте (MIT) в начале 80-х гг. прошлого века. Версия 10 системы X Window System, вышедшая в 1985 г., была первой версией, которой удалось достичь широкого распространения, а вскоре после нее вышла версия 11 (X11). Благодаря относительно либеральным условиям лицензионного соглашения, систему X стали переносить на другие платформы, появились многие варианты ее реализации.

В 1998 г. группа X Consortium, входящая в состав MIT, выработала общее направление развития протокола X. В течение последующих десяти лет эта группа, а также ее последователи, постоянно работали над обновлениями протокола. Самой последней и большой версией является X11R7.1 — теперь в номерах версий добавляются новые числа, а не увеличиваются существующие.

Версия XFree86 являлась фактической реализацией X-сервера для Linux (а также многих других платформ) до тех пор, пока в 2004 г. не изменились условия лицензионного соглашения, вследствие чего многие дистрибьюторы были вынуждены переключиться на ветвь XFree86, которая не подпадала под условия нового лицензионного соглашения. Эта ветвь поддерживается некоммерческой организацией X.Org Foundation и сегодня является доминирующей реализацией для Linux. Кроме того, версию X.Org Server можно переносить на платформу Windows для использования в среде совместимо-

сти Cygwin Linux. (Для Windows существует несколько коммерческих версий X-сервера; более подробно об этом можно прочитать в разделе 26.2.)

В этой главе будет рассмотрена версия X.Org, которую мы использовали во всех наших образцах дистрибутивов. Реализации версий X.Org и XFree86 имеют расхождения в плане архитектуры, однако многие административные функции является одинаковыми. Чтобы использовать команды для версии XFree86, можно менять в них записи "xorg" на "xf86". С каждым днем версия XFree86 все больше устаревает, поэтому мы о ней говорить не будем.

В системе X Window System можно выделить несколько ключевых компонентов. Во-первых, она содержит *диспетчер дисплеев* (display manager), главной задачей которого является аутентификация пользователей, их регистрация и запуск исходной среды для сценариев запуска системы. Диспетчер дисплеев запускает *X-сервер* (X server), который определяет абстрактный интерфейс для растровых изображений и устройств ввода (например, для клавиатуры и мыши). Сценарии запуска запускают *диспетчер окон* (window manager), который позволяет пользователям перемещать, сворачивать, восстанавливать и изменять размеры окон, а также управлять отдельными виртуальными рабочими столами. Наконец, на самом нижнем уровне приложения связываются с *библиотекой графических интерфейсных элементов* (widget library), которая реализует механизмы высокоуровневого пользовательского интерфейса, такие как кнопки и меню. На рис. 22.1 можно проследить взаимосвязь между диспетчером дисплеев, X-сервером и приложениями клиентов.

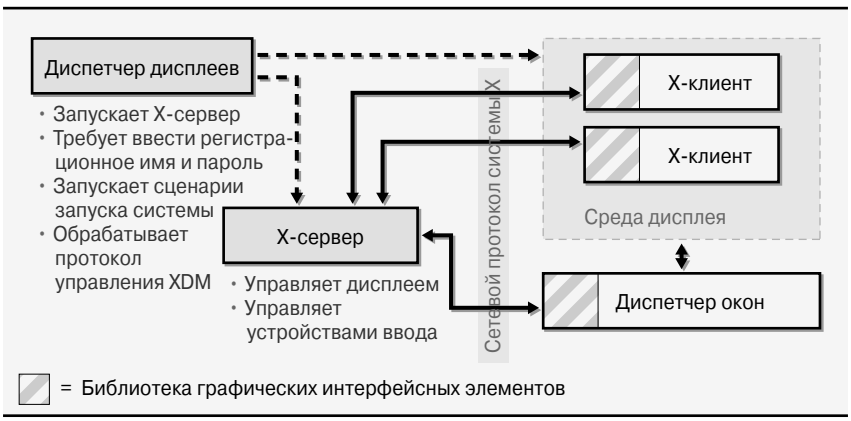


Рис. 22.1. Модель клиент/сервер в системе X

X-сервер понимает только базовый набор рисуемых примитивов в сетевом API; он не определяет программный интерфейс для высокоуровневых объектов, таких как кнопки, текстовые окна, меню и ползунки. Это обусловлено двумя причинами. Во-первых, это позволяет X-серверу работать на совершенно другом компьютере из приложения клиента. Во-вторых, X-сервер позволит серверу поддерживать разнообразные диспетчеры окон и наборы библиотек графических интерфейсных элементов.

У разработчиков приложений имеются свои библиотеки графических интерфейсных элементов и стандарты пользовательских интерфейсов. К сожалению, выбор часто зависит от "религиозной" приверженности, а не от того, какие задачи нужно решать разработчикам в действительности. Несмотря на удобства, которые открываются благодаря

свободе выбора, агностицизм пользовательского интерфейса системы X в течение долгих лет сдерживал разработку качественных пользовательских интерфейсов.

В этой главе мы поговорим о том, как запускаются программы на дистанционном дисплее и как производится процесс аутентификации. Мы рассмотрим также вопрос конфигурирования сервера X.Org и поиска и устранения ошибок в конфигурации. В конце главы мы вкратце поговорим о некоторых доступных диспетчерах окон и средах рабочего стола.

22.1. Диспетчер дисплеев

Диспетчер дисплеев является первой утилитой, с которой сталкивается пользователь, садясь за компьютер. Он не является обязательным; многие пользователи отключают эту утилиту и запускают систему X из текстовой консоли или из сценария `.login`, выполняя команду `startx` (она является упаковщиком программы `xinit`, которая запускает X-сервер).

С другой стороны, диспетчер дисплеев показывает пользователю привлекательный и дружественный экран регистрации, и предлагает возможность вносить изменения в конфигурацию. Диспетчер дисплеев может позволить выполнять дистанционную регистрацию на других X-серверах посредством протокола XDMCP. Он может также поддерживать аутентификацию дисплея (см. раздел “Аутентификация клиентов” далее в этой главе). Исходный диспетчер дисплеев называется `xdm` (для диспетчера дисплеев системы X), однако современные варианты, такие как `gdm` (диспетчер дисплеев для GNOME) и `kdm` (диспетчер дисплеев для KDE) предлагают практически такой же набор функций и имеют более привлекательный вид.

Как правило, диспетчер дисплеев запускает X-сервер, осуществляет аутентификацию пользователей, вводит пользователей в систему и выполняет сценарии запуска пользователей. Набор конфигурационных файлов, которые чаще всего находятся в каталоге `/etc/X11/xdm`, определяет, как будет работать утилита `xdm`. К примеру, файл `Xservers` можно отредактировать таким образом, чтобы изменить номер дисплея, используемого для данного сервера, если на других виртуальных терминалах будет работать множество серверов. Или же можно изменить компоновку сервера с помощью опции `-layout`, если вы определили компоновку, подходящую для множества систем.

После запуска X-сервера утилита `xdm` предлагает ввести имя пользователя и пароль. Пароль пользователя проверяется в соответствии с модулями PAM (Pluggable Authentication Modules — подключаемые модули аутентификации), которые определены в файле `.etc/pam.d/xdm` (или `kdm/gdm`, если вы используете диспетчеры дисплеев KDE или GNOME). На экране регистрации будет предложена опция, позволяющая зарегистрироваться в нескольких альтернативных средах рабочего стола, включая важную безотказную опцию, о которой мы поговорим немного позже.

📖 Подробные сведения о PAM можно найти в конце раздела 20.5.

Последней обязанностью диспетчера дисплеев является выполнение сценария интерпретатора команд `XSession`, который настраивает среду рабочего стола пользователя. Сценарий `XSession`, который, как правило, находится в каталоге `/etc/X11/xdm`, является общесистемным сценарием запуска. Он задает стандартные настройки приложений, инсталлирует стандартные клавиатурные привязки и выбирает языковые настройки. Затем сценарий `XSession` выполняет личный пользовательский сценарий запуска, который обычно называется `~/.xsession`. Он запускает диспетчер дисплеев, панель

задач, вспомогательные аплеты, а также может запускать и другие программы. GNOME и KDE имеют собственные сценарии запуска, которые конфигурируют рабочий стол пользователя в соответствии со средствами конфигурации GNOME и KDE. При этой схеме ошибок возникает меньше, чем при редактировании пользователями собственных сценариев запуска.

Когда работа сценария `~/.xsession` будет завершена, пользователь будет выведен из системы, и диспетчер дисплеев снова предложит ввести имя пользователя и пароль. Таким образом, сценарий `~/.xsession` должен запускать все программы в фоновом режиме (добавляя символ `&` в конце каждой команды), *кроме самой последней*, которой обычно является диспетчер окон. (Если все команды в сценарии `~/.xsession` будут работать в фоновом режиме, работа сценария будет тут же прекращена, и пользователь будет выведен из системы сразу же после входа в нее.) Если же диспетчер окон будет работать в качестве приоритетного процесса, пользователь сможет выйти из системы только после того, как будет завершена работа диспетчера окон.

Безотказная опция позволяет пользователям входить в систему с целью устранения сбоев в проблемных сценариях запуска. Как правило, эту опцию можно выбирать на экране регистрации диспетчера дисплеев. В ее случае открывается только простое окно терминала; после закрытия окна пользователь выходит из системы. Благодаря этому пользователь может самостоятельно устранить возникшие неполадки, не обращаясь за помощью к администратору.

Как правило, самой распространенной проблемой при запуске системы является как раз то, что процессу не присваивается высокий уровень приоритета, и он остается работать в фоне. Однако это не единственная проблема, которая может возникнуть. Если причину возникновения проблемы установить сложно, можно просмотреть файл `~/.xsession-errors`, в котором содержатся результаты выполнения команд в сценарии `~/.xsession`. Постарайтесь найти в этом файле ошибки или какое-либо необычное поведение системы. Если и это не поможет, переместите сценарий `~/.xsession` в какое-нибудь изолированное место и попробуйте войти в систему без него. Затем постепенно восстанавливайте одну-две строки до тех пор, пока не будет найдена “проблемная” строка.

22.2. Запуск X-приложения

Процедура, необходимая для запуска X-приложения, на первый взгляд может показаться чересчур сложной. Тем не менее, вскоре вы обнаружите, что клиент-серверная модель дисплеев предлагает высокую степень гибкости. Так как информация об обновлении дисплея передается по сети, приложение (клиент) может работать на совершенно другом компьютере, а не на том, где оно отображает свой графический интерфейс пользователя (на сервере). X-сервер может соединяться со многими другими приложениями, каждое из которых будет работать на отдельном компьютере.

Чтобы эта схема могла работать, клиенты должны знать, к какому дисплею нужно подключаться, и какой экран нужно занимать на этом дисплее. После соединения с дисплеем клиенты должны пройти аутентификацию на X-сервере, гарантируя, что человек, сидящий перед дисплеем, имеет право устанавливать соединение.

Тем не менее, даже после прохождения аутентификации система X не обязательно будет полностью защищена. Еще один безопасный способ управления соединения заключается в использовании протокола SSH (см. раздел “Перенаправление соединений с помощью протокола SSH” далее в этой главе). Мы настоятельно рекомендуем исполь-

зовать протокол SSH для соединений системы X с сетью Internet. В любом случае, это будет нелишним для локальной передачи данных.

📖 Подробные сведения о протоколе SSH даны в разделе 20.11.

Переменная окружения DISPLAY

X-приложения обращаются к переменной окружения DISPLAY для того, чтобы узнать, где они будут отображать свои данные. Эта переменная содержит имя или IP-адрес сервера, номер дисплея (он указывает на определенный экземпляр X-сервера, с которым нужно соединиться) и необязательный номер экрана (для дисплеев с несколькими мониторами). Когда приложения работают на том же компьютере, на котором отображаются их интерфейсы, для простоты записи большинство этих параметров можно опустить.

Следующий пример показывает и формат информации дисплея, и синтаксис интерпретатора команд bash, используемого для задания переменной окружения:

```
client$ DISPLAY=имя_сервера.домен.com:10.2; export DISPLAY
```

Эта запись расшифровывается так: X-приложение находится по адресу *имя_сервера.домен*.com, дисплей 10, экран 2. Приложения устанавливают TCP-соединение с сервером на порту с номером 6000 плюс номер дисплея (в данном примере это порт 6010), на котором X-сервер, обрабатывающий данный дисплей, должен вести прослушивание.

Следует иметь в виду, что каждый процесс имеет свои собственные переменные окружения. Если переменную окружения DISPLAY задать для интерпретатора команд (оболочки), то ее значение будет передано только тем программам, которые будут запускаться этим интерпретатором. Если выполнить вышеприведенные команды в одной программе xterm, а затем попытаться запустить любимое X-приложение в другой программе, то приложение не будет иметь доступа к созданной вами переменной DISPLAY.

Обратить внимание нужно и на другой момент. Дело в том, что хотя X-приложения посылают свои графические выходные данные на обозначенный X-сервер, они попрежнему имеют локальные каналы stdout и stderr. На терминальное окно, из которого было запущено X-приложение, могут направляться некоторые выходные данные об ошибках.

Если и клиент, и сервер находятся в вашей организации, вы можете убрать из переменной DISPLAY полное доменное имя сервера, в зависимости от того, каким образом был сконфигурирован распознаватель вашего сервера имен. Также, поскольку большинство систем работают на одном X-сервере, дисплей обычно имеет нулевой номер. Номер экрана можно опустить, в результате чего будет подразумеваться номер 0. Итак, чаще всего удобнее будет присваивать переменной DISPLAY значение *имя_сервера*:*0*.

📖 О конфигурировании распознавателя DNS рассказывалось в разделе 15.10.

Если приложение клиента будет работать на том же компьютере, что и X-сервер, переменную DISPLAY можно еще больше упростить, опустив имя компьютера. Эта особенность не просто косметическая: если указать пустое имя, то для связи с X-сервером библиотеки клиента будут использовать сокет домена UNIX, а не сокет сети. Кроме того, что этот тип соединения является более быстрым и эффективным, он позволяет обойти любые ограничения, вводимые брандмауэром в локальной системе, которые блокируют внешние соединения с системой X. Наиболее простым из возможных значений переменной окружения DISPLAY является нулевое значение (:0).

Библиотеки клиента, которые обращаются к переменной окружения DISPLAY, обычно принимают эту информацию также и в виде аргумента командной строки. Например, команда

```
client$ xprogram -display имя_сервера:0
```

эквивалентна запуску программы с помощью переменной DISPLAY, имеющей значение *имя_сервера*:0. Опции командной строки отменяют значения переменной окружения. Эта особенность будет очень удобна, если на одном и том же компьютере вы будете запускать несколько программ, обрабатываемых разными дисплеями.

Аутентификация клиентов

Несмотря на то что X-среда обычно считается относительно незащищенной, любая мера предосторожности поможет защититься от несанкционированного доступа. До того времени, как вопрос безопасности системы стал крайне важным, X-серверы принимали соединения от любого клиента, работающего на компьютере, который помечался как безопасный и с помощью команды xhost. Однако в связи с тем, что любой пользователь, работающий на этом компьютере, мог соединиться с вашим дисплеем и причинить вред (случайно или преднамеренно), от метода предоставления доступа посредством xhost отказались навсегда. Больше о нем и мы не будем упоминать.

Наиболее распространенным альтернативным вариантом защиты является аутентификация с использованием "магических" cookie-наборов (magic cookie). Если вы знакомы с работой в сети Internet, то cookie-наборы вам тоже должны быть знакомы; в данном контексте cookie-наборы используются для аутентификации X-соединений. Суть идеи заключается в том, что диспетчер дисплеев X в самом начале процедуры регистрации в системе генерирует большое случайное число, которые и называется cookie-набором. Cookie-набор сервера записывается в файл ~/.Xauthority, который находится в домашнем каталоге пользователя. Соединение могут выполнять любые клиенты, которым известен cookie-набор. Пользователи могут выполнять команду xauth для того, чтобы просматривать существующие cookie-наборы и добавлять в этот файл свои собственные cookie-наборы.

Как работает эта схема, можно проще всего продемонстрировать на примере. Предположим, что вы задали вашу переменную DISPLAY в системе клиента для отображения X-приложений на компьютере, за которым вы работаете. Однако когда вы запускаете программу, на экран выводится ошибка, которая выглядит примерно так:

```
client$ xprogram -display server:0
Xlib: connection to "server:0.0" refused by server
xprogram: unable to open display 'server:0'
```

Это сообщение говорит о том, что у клиента нет подходящего cookie-набора, поэтому дистанционный сервер отклонил данное соединение. Чтобы получить подходящий cookie-набор, нужно зарегистрироваться на сервере (это уже может быть сделано, если вы пытаетесь отобразить программу прямо на нем) и вывести список cookie-наборов сервера, выполнив команду xauth list:

```
server$ xauth list
server:0 MIT-MAGIC-COOKIE-1 f9d888df6077819ef4d788fab778dc9f
server/unix:0 MIT-MAGIC-COOKIE-1 f9d888df6077819ef4d788fab778dc9f
localhost:0 MIT-MAGIC-COOKIE-1 cb6cbf9e5c24128749feddd47f0e0779
```

Каждый сетевой интерфейс на сервере имеет свою запись. В данном примере мы имеем cookie-набор для Ethernet, cookie-набор для сокета домена UNIX, который используется для локальных соединений, и cookie-набор для сетевого интерфейса обратной связи localhost.

Самый простой способ обзавестись cookie-набором на клиенте (при условии, что не используется сетевой протокол SSH, который сам заботится о cookie-наборах) заключается в применении знакомой всем операции вырезания и вставки. Многие эмуляторы терминалов (например, `xterm`) позволяют выделять текст с помощью мыши и вставлять его в другом окне, обычно посредством щелчка средней кнопки мыши. В этом случае удобно то, что команда `xauth add` принимает в качестве ввода тот же формат, который отображает команда `xauth list`. Добавить cookie-набор к клиенту можно следующим образом:

```
client$ xauth add server:0 MIT-MAGIC-COOKIE-1
  9d888df6077819ef4d788fab778dc9f
```

Чтобы убедиться в том, что cookie-набор был добавлен правильно, нужно выполнить команду `xauth list` на клиенте. Если переменная окружения `DISPLAY` будет задана, и будет добавлен действительный "магический" cookie-набор, приложения смогут отображать свои данные на сервере.

Если у вас возникнут проблемы с работой cookie-наборов, можно на время вернуться к процедуре аутентификации с использованием команды `xhost` — просто для того, чтобы посмотреть, не возникают ли какие-то проблемы в ней (например, ограничения, выдвигаемые брандмауэрами или локальной сетью, которые не предоставляют клиенту доступ к серверу). Чтобы отключить аутентификацию `xhost` после выполненного вами теста, нужно выполнить команду `xhost -` (то есть, `xhost` со знаком дефиса в качестве единственного аргумента).

Перенаправление соединений с помощью протокола SSH

"Магические" cookie-наборы усиливают защиту, однако использовать одни лишь cookie-наборы недостаточно. Любой пользователь, который может получить cookie-набор вашего дисплея, сможет подключиться к нему и запустить программы, которые будут отслеживать ваши действия. Даже без вашего cookie-набора X-протокол будет передавать данные по сети без шифрования, в результате чего буквально каждый пользователь сможет их перехватывать.

Усилить защиту можно с помощью протокола SSH (secure shell — защищенный интерпретатор команд). Этот протокол обеспечивает аутентификацию и шифрование на терминале. Тем не менее, SSH может также передавать произвольные сетевые данные, включая данные протокола X, по защищенному каналу. Механизм перенаправления, используемый в системе X, подобен обобщенному перенаправлению портов с помощью SSH, однако поскольку SSH знает об особенностях системы X, вы получаете дополнительные преимущества, включая псевдодисплей на дистанционном компьютере и "договорную" передачу "магических" cookie-наборов.

📖 Подробные сведения о протоколе SSH можно найти в разделе 20.11.

Протокол SSH обычно используется для передачи данных между компьютером, на котором работает X-сервер, и компьютером, на котором функционируют X-программы. Эта схема может привести читателя в замешательство, поскольку *SSH-клиент* работает

на том же компьютере, что и *X-сервер*, и соединяется он с *SSH-сервером*, который работает на том же компьютере, где работают приложения *X-клиента*. Ситуация усугубляется еще и тем, что виртуальный дисплей, который формирует SSH для вашего X-сервера, является локальным по отношению к дистанционной системе. На рис. 22.2 показано, как осуществляется передача данных посредством SSH-соединения в системе X.

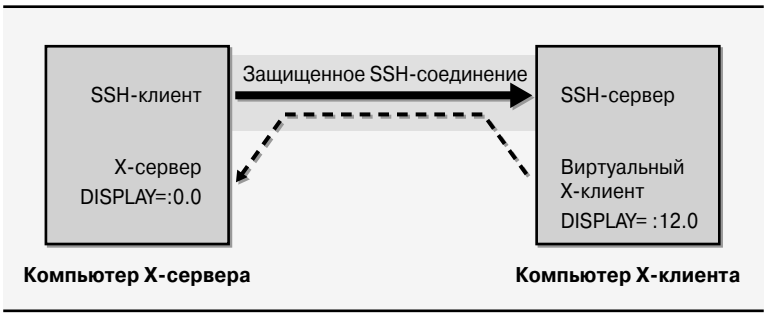


Рис. 22.2. Использование протокола SSH в системе X

Вашу переменную окружения `DISPLAY` и информацию об аутентификации можно задать автоматически посредством команды `ssh`. Номер дисплея начинается с `:10.0` и увеличивается на единицу для каждого SSH-соединения, при котором передаются данные в системе X.

В качестве примера можно предложить следующую последовательность:

```
x-server$ ssh -v -X x-client.mydomain.com
OpenSSH_3.9p1, OpenSSL 0.9.7a Feb 19 2003
debug1: Reading configuration data /home/boggs/.ssh/config
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to x-client.mydomain.com [192.168.15.9] port 22.
debug1: Connection established.
Enter passphrase for key '/home/boggs/.ssh/id_rsa':
debug1: read PEM private key done: type RSA
debug1: Authentication succeeded (publickey).
debug1: Entering interactive session.
debug1: Requesting X11 forwarding with authentication spoofing.
debug1: Requesting authentication agent forwarding.
x-client$
```

В последних двух строчках видно, что клиент получает запрос на перенаправление данных для X11-приложений. Перенаправление в системе X может быть разрешено как на SSH-сервере, так и на SSH-клиенте; кроме того, у клиента должен быть подходящий cookie-набор для сервера. Если у вас не получается организовать перенаправление, попробуйте использовать флаги `-X` и `-v`, как было показано выше (для OpenSSH), чтобы явным образом разрешить перенаправление в системе X и запросить подробные выходные данные. Также можно просмотреть глобальные конфигурационные файлы SSH в каталоге `/etc/ssh` и убедиться в том, что перенаправление в системе X11 не было отменено администратором. Сразу после входа в систему вы можете проверить ваш дисплей и магические cookie-наборы:

```
x-client$ echo $DISPLAY
localhost:12.0
x-client$ xauth list
x-client/unix:12 MIT-MAGIC-COOKIE-1 a54b67121eb94c8a807f3ab0a67a51f2
```

Обратите внимание на то, что переменная DISPLAY указывает на виртуальный дисплей на SSH-сервере. Остальным SSH-соединениям (производимым как вами, так и другими пользователями) присваиваются разные виртуальные номера дисплеев. Если переменная DISPLAY и cookie-набор будут заданы правильно, вы сможете запустить приложение клиента.

```
x-client$ xeyes
debug1: client_input_channel_open: ctype x11 rchan 4 win 65536 max 16384
debug1: client_request_x11: request from 127.0.0.1 35411
debug1: channel 1: new [x11]
debug1: confirm x11
debug1: channel 1: FORCE input drain
```

Если с помощью команды ssh -v разрешить показ отладочной информации, вы увидите, что ssh получила запрос на X-соединение и покорно перенаправила его X-серверу. Сам процесс перенаправления может производиться немного медленно, однако приложение в конечно счете должно появиться на вашем экране.

22.3. Конфигурирование X-сервера

Сервер X.Org, или Xorg сложно поддается конфигурированию в данной среде оборудования. И это небезосновательно. Отчасти сложность конфигурирования Xorg объясняется широким разнообразием оборудования для графики, устройств ввода, режимов видео, разрешений и глубины цвета, которую поддерживают устройства. Во времена системы XFree86 новых пользователей часто ошеломлял зашифрованный конфигурационный файл, в котором содержалась информация, напоминающая набор случайных чисел, взятых из несуществующего руководства к монитору. В конфигурационных файлах Xorg эти числа стали более структурированными.

Конфигурационный файл Xorg обычно находится в каталоге /etc/X11/xorg.conf, однако X-сервер будет просматривать целые кипы каталогов, пытаясь найти его. Полный список есть на man-странице, однако нужно иметь в виду, что некоторые пути, по которым Xorg ведет поиск, содержат имя компьютера и глобальную переменную, что облегчает хранение конфигурационных файлов для множества систем в одном месте.

Некоторые программы могут помочь вам в конфигурировании системы X (например, xorgconfig), однако вам желательно самим разобраться со структурой конфигурационного файла на тот случай, если вы захотите самостоятельно просматривать или редактировать эти файлы. Первоначальную вспомогательную информацию по этому вопросу можно найти прямо на X-сервере, выполнив команду Xorg -probeonly и просмотрев выходные данные для видеоплаты и прочие тестовые значения. Если выполнить команду Xorg -configure, X-сервер создаст исходный конфигурационный файл, который будет основан на тестовых значениях. Для начала этой информации вам вполне хватит.

Файл xorg.conf состоит из нескольких секций, каждая из которых начинается с ключевого слова Section и заканчивается ключевым словом EndSection. Наиболее распространенные типы секций перечислены в табл. 22.1.

Таблица 22.1. Секции файла `xorg.conf`

Секция	Описание
`ServerFlags`	Перечисляет главные конфигурационные параметры X-сервера
`Module`	Определяет динамически загружаемые расширения для ускоренной графики, визуализаторов шрифтов и т.п.
`Device`	Конфигурирует информацию о видеоплате, драйвере и оборудовании
`Monitor`	Описывает физические параметры монитора, включая синхронизацию и разрешения дисплеев
`Screen`	Связывает монитор с видеоплатой (`Device`) и определяет разрешения и глубину цвета, доступных в данной конфигурации
`InputDevice`	Определяет устройства ввода, такие как клавиатуры и мыши
`ServerLayout`	Связывает устройства ввода с экранами и позиционирует экраны относительно друг друга

Как правило, создавать конфигурационный файл проще всего “снизу вверх”, начиная с определения секций для устройств ввода и вывода, и комбинируя их в различных компоновках. При таком иерархическом подходе простой конфигурационный файл можно использовать для многих X-серверов, каждый с разным оборудованием. Этот подход будет удобен и для одной системы, в которой установлено несколько видеоплат и используется несколько мониторов.

На рис. 22.3 показано, как некоторые из этих секций располагаются в иерархии конфигурации X.Org. Физический дисплей `Monitor` и видеоплата `Device` формируют экран `Screen`. Набор экранов `Screen` плюс устройства ввода `InputDevices` формируют компоновку `ServerLayout`. В конфигурационном файле может быть определено множество компоновок сервера, однако только одна из них будет активной в данном экземпляре процесса `Xorg`.

Некоторые из этих секций, составляющие файл `xorg.conf`, являются относительно фиксированными. Стандартные секции часто могут браться прямо из существующего конфигурационного файла или из его образца. Другие (например, `Device`, `Monitor`, `Screen`, `InputDevice` и `ServerLayout`) зависят от того, как настроено оборудование компьютера. В следующих подразделах мы рассмотрим наиболее интересные из этих секций.

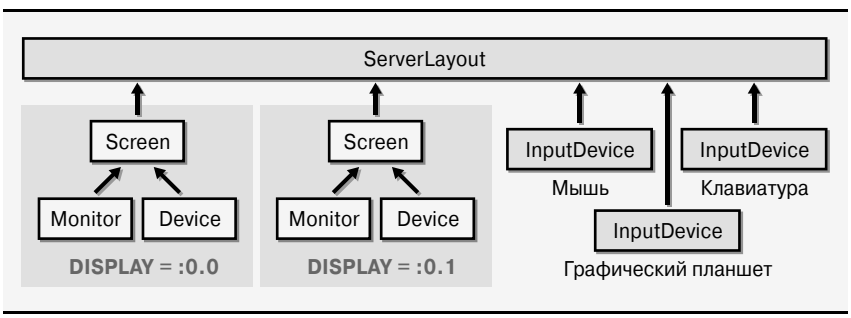


Рис. 22.3. Взаимоотношение между секциями конфигурации `xorg.conf`

Секция Device

Секция Device описывает определенную видеоплату. Вы должны указать строку, которая идентифицирует плату и драйвер, подходящий для данного устройства. Драйвер загружается только в том случае, если на это устройство имеется ссылка в соответствующей секции Screen. Типичная секция Device может выглядеть примерно так:

```
Section "Device"
    Identifier      "STBVirge3D"
    Driver          "s3virge"
    опция           значение
    ...
EndSection
```

На странице руководства этого драйвера (в данном случае это s3virge) перечисляется и описывается оборудование, которым он управляет, и опции, которые он поддерживает. Если во время работы возникают какие-то странные явления, можно отключить аппаратное ускорение (если эта функция поддерживается), сократив доступ к видеопамяти, или модифицировать параметры PCI. Однако перед тем как что-либо изменять, желательно проконсультироваться в сети Web или у специалиста.

Секция Monitor

В секции Monitor описываются дисплеи, подключенные к вашему компьютеру. В ней могут быть определены детализированные значения синхронизации. Информация о синхронизации необходима для оборудования старого образца; тем не менее, она может быть интересна и многим современным мониторам. Спецификации дисплеев обычно можно получить на Web-сайтах производителей, однако самым лучшим источником, естественно, является руководство, прилагаемое к монитору. В любом случае, вам нужно будет знать как минимум частоты горизонтальной и вертикальной развертки для вашей модели.

Типичная секция Monitor выглядит примерно так:

```
Section "Monitor"
    Identifier      "ViewSonic"
    Option          "DPMS"
    HorizSync       30-65
    VertRefresh     50-120
EndSection
```

Как и во всех секциях, строка Identifier присваивает имя, с помощью которого вы будете ссылаться на данный монитор. Здесь мы включили функцию DPMS (Display Power Management Signaling — сигналы управления энергопотреблением дисплеев), благодаря которой X-сервер будет выключать монитор, когда мы будем делать перерыв на чашечку кофе или просто отлучаться на некоторое время.

Строки HorizSync и VertRefresh должны содержать значения, подходящие для вашего монитора. Они могут быть определены в виде диапазона частот (как это сделано в приведенном выше примере), или в виде набора дискретных значений, разделяемых запятыми. Драйвер теоретически может протестировать поддерживаемые режимы, однако если параметры будут определены, он не будет использовать неподдерживаемые частоты.

В первых мониторах с электронно-лучевой трубкой (CRT) сигналы, излучаемые на неподдерживаемой частоте, выводили мониторы из строя; современные мониторы этого типа имеют более устойчивые характеристики. Современные мониторы с жидкокристаллическими панелями (LCD) являются более совершенными устройствами, однако экспериментировать с частотами монитора все равно следует осторожно. Вы должны быть готовы к тому, что монитор потухнет, если он получит сигнал, не поддерживаемый им.

Секция Screen

В секции `Screen` устройство (видеоплата) связывается с монитором при определенной глубине цвета, а также задаются параметры разрешения дисплея. Ниже показан пример, в котором используются упомянутые видеоплата и монитор.

```
Section "Screen"
    Identifier      "Screen 2"
    Device          "STBVirge3D"
    Monitor         "ViewSonic"
    DefaultDepth    24
    Subsection      "Display"
       Depth        8
       Modes        "640x400"
    EndSubsection
    Subsection      "Display"
       Depth        16
       Modes        "640x400" "640x480" "800x600" "1024x768"
    EndSubsection
    Subsection      "Display"
       Depth        24
       Modes        "1280x1024" "1024x768" "800x600" "640x400" "640x480"
    EndSubsection
EndSection
```

Как можно было предположить, в строке `Identifier` указывается имя экрана, и перечисляются идентификаторы ранее определенной видеоплаты и монитора. Это первая представленная нами секция, в которой имеются подсекции. Для каждой глубины цвета определяется одна подсекция, а стандартное значение глубины задается в поле `DefaultDepth`.

Данный экземпляр X-сервера может работать только при одной глубине цвета. В самом начале сервер определяет, какие разрешения поддерживаются для данной глубины цвета. Возможные разрешения зависят, как правило, от количества памяти, имеющейся на видеоплате. В старых видеоплатах, имеющих небольшой объем памяти, разрешение было ограничено высокими значениями глубины цвета. В разделе “Специальные комбинации клавиш в системе X” этой главы описано, в каком порядке производится выбор определенных здесь разрешений.

Любая приличная современная видеоплата должна уметь работать с вашим монитором с полным его разрешением при 24- или 32-битном цвете. Если вам понадобится запустить старые программы, для которых сервер должен работать с 8-битным цветом, запустите другой X-сервер в отдельной виртуальной консоли. Чтобы отменить действие опции `DefaultDepth`, в командной строке `Xorg` используйте флаг `-depth 8`.

Секция InputDevice

В секции InputDevice описывается устройство ввода (например, клавиатура или мышь). Каждое устройство получает свою собственную секцию InputDevice и, как и в других секциях, каждое устройство получает имя в поле Identifier. Если вы используете один конфигурационный файл для нескольких компьютеров с разным оборудованием, вы можете определить все устройства ввода; использоваться будут только те из них, которые будут указаны в секции ServerLayout. Ниже показано типичное определение клавиатуры:

```
Section "InputDevice"
    Identifier      "Generic Keyboard"
    Driver          "Keyboard"
    Option          "AutoRepeat" "500 30"
    Option          "XkbModel" "pc104"
    Option          "XkbLayout" "us"
EndSection
```

Среди прочего, в определении клавиатуры можно задать такие опции, которые определят подходящее положение клавиш <Control> и <Caps Lock>. В данном примере опция AutoRepeat определяет, насколько долго нужно удерживать клавишу нажатой, чтобы это привело к повтору нажатия символа, и как быстро будет повторяться ввод символа.

Конфигурация мыши определяется в другой секции InputDevice:

```
Section "InputDevice"
    Identifier      "Generic Mouse"
    Driver          "mouse"
    Option          "CorePointer"
    Option          "Device" "/dev/input/mice"
    Option          "Protocol" "IMPS/2"
    Option          "Emulate3Buttons" "off"
    Option          "ZAxisMapping" "4 5"
EndSection
```

Опция CorePointer обозначает эту мышь как основное указывающее устройство в системе. Файл устройства (Device), связанный с мышью, определен в поле Option; обычно это файл /dev/input/mice, который является мультиплексором мыши. Протокол (Protocol) зависит от конкретного производителя мыши; если ему присвоить значение auto, сервер попытается определить его самостоятельно. Если колесико мыши не работает, попробуйте воспользоваться протоколом IMPS/2. Если мышь содержит много кнопок (больше 3), используйте протокол ExplorerPS/2.

Если /dev/input/mice не подходит для вашей мыши, значит, конфигурация является более сложной. Программа gpm реализует средства вырезания и вставки, выполняемые с помощью мыши, в текстовых виртуальных терминалах. Работать с традиционной мышью одновременно может только одна программа. Чтобы избежать этой проблемы, qpm воспроизводит данные мыши в файле FIFO, поэтому приложения, такие как X-сервер, тоже могут их видеть (используя файл FIFO в качестве устройства мыши)[1]. Такая схема позволяет добавить функции gpm в систему, оставляя ее при этом относительно прозрачной.

[1] Файлы FIFO создаются посредством команды mknod. Например, mknod p /dev/gpmdata.

Например, благодаря следующей команде программа gpm получит входные данные из /dev/mouse с помощью протокола IMPS/2 и перенаправит их в файл FIFO /dev/gpmdata (имя не конфигурируется) без перевода в протокол.

```
$ gpm -m /dev/mouse -t imps2 -Rraw
```

Затем вам нужно будет изменить устройство мыши на /dev/gpmdata в файле xorg.conf. Так как gpm нужно запускать до начала работы X-сервера, эту команду необходимо поместить в сценарий запуска системы, такой как /etc/init.d/gpm. О сценариях запуска системы подробно рассказано в разделе 2.5.

Опция Emulate3Button позволяет двухкнопочным мышам имитировать работу трехкнопочных, определяя щелчок двумя кнопками как щелчок средней кнопкой мыши. Опция ZAxisMapping иногда нужна для поддержки прокрутки колесика или работы джойстика; эта поддержка осуществляется путем сопоставления кнопок. Большинство современных мышей имеют как минимум три кнопки, колесико прокрутки, встроенный MP3-плеер, щетку для массажа ступней и холодильник для пива[2].

Секция ServerLayout

Секция ServerLayout является узлом верхнего уровня в иерархии конфигурации. Каждая аппаратная конфигурация, на которой будет работать сервер, должна иметь свой собственный экземпляр секции ServerLayout. Компоновка, используемая определенным X-сервером, обычно определяется в командной строке сервера.

Ниже показан пример всей секции ServerLayout.

```
Section "ServerLayout"
    Identifier    "Simple Layout"
    Screen        "Screen 1" LeftOf "Screen 2"
    Screen        "Screen 2" RightOf "Screen 1"
    InputDevice   "Generic Mouse" "CorePointer"
    InputDevice   "Generic Keyboard" "CoreKeyboard"
    Option        "BlankTime" "10" # Погасить экран через 10 минут
    Option        "StandbyTime" "20" # Отключить экран через 20 минут (DPMS)
    Option        "SuspendTime" "60" # Полное засыпание через 60 минут (DPMS)
    Option        "OffTime" "120"# Отключить DPMS-монитор через 2 часа
EndSection
```

Эта секция связывает вместе все остальные секции, представляя X-дисплей. Она начинается с необходимого поля Identifier, в котором указывается данная конкретная компоновка. После этого набор экранов связывается с компоновкой[3]. Если к отдельным видеоплатам подключено несколько мониторов, то каждый экран будет определяться вместе с необязательными направлениями, чтобы показать из физическое месторасположение. В данном примере один экран (Screen 1) находится слева, а другой (Screen 2) — справа.

Некоторые видеоплаты могут управлять одновременно несколькими мониторами. В данном случае в секции ServerLayout определен только один экран Screen. Для видеоплат производства NVIDIA, которые в настоящее время чаще всего используются для данных целей в среде Linux, в секции Driver нужно задать опцию для обозначения под-

[2] Xorg поддерживает не все опции. Некоторые продаются отдельно.

[3] Помните, что экраны идентифицируют комбинацию монитор/видеоплата при определенной глубине цвета.

держки режима TwinView. Подробности этой конфигурации выходят за рамки этой книги, однако соответствующую информацию можно найти на различных Web-форумах.

Вслед за списком экранов следует набор устройств ввода, которые необходимо связать с этой компоновкой. Опции CorePointer и CoreKeyboard переходят в секцию InputDevice, чтобы показать, что эти устройства должны быть активными в данной конфигурации. Эти опции могут быть заданы прямо в соответствующих секциях InputDevice, однако проще всего задавать их в секции ServerLayout.

В последних нескольких строках конфигурируется несколько опций, специфических для компоновки. В вышеприведенном примере все они связаны с DMPS — интерфейсом, благодаря которому мониторы, соответствующие стандарту Energy Star, знают, когда нужно отключать питание. Мониторы должны также иметь собственные опции DMPS, разрешенные в соответствующих секциях Monitor.

22.4. Устранение неполадок и отладка

Конфигурация X-сервера прошла долгий путь за последние десять лет, однако она по-прежнему остается сложной, и вам будет непросто добиться того, чтобы все работало именно так, как вы того хотите. Вам придется экспериментировать с частотами мониторов, опциями драйверов, оригинальными драйверами или расширениями для трехмерной визуализации. По иронии, сбои в работе мониторов возникают именно тогда, когда вам крайне необходимо просмотреть информацию об отладке на экране. К счастью, сервер X.Org может дать вам любую необходимую информацию (и даже сверх того), которая поможет вам выявить причину возникновения проблемы.

Специальные комбинации клавиш в системе X

Так как X-сервер принимает на себя управление вашей клавиатурой, дисплеем и мышью, вы можете себе представить, что он может оставить вам небольшое количество ресурсов, и выключит питание системы, если что-то не будет работать. Однако прежде чем дело дойдет до этого, нужно попытаться выполнить некоторые действия. Если нажать и удерживать клавиши <Control> и <Alt>, а затем нажать функциональную клавишу (<F1> — <F6>), X-сервер переключит вас на один из текстовых виртуальных терминалов. В нем вы сможете войти в систему и произвести отладку. Чтобы вернуться к X-серверу, работающему на терминале 7, нужно нажать <Alt+F7>[4]. Если вы работаете в сети, вы можете также попробовать войти в систему на другом компьютере, чтобы уничтожить X-сервер, прежде чем нажимать кнопку сброса.

Если монитор не синхронизируется с видеосигналом, поступающим с видеоплаты, попытайтесь изменить разрешение экрана. Доступные разрешения указаны в строке Modes секции Screen конфигурационного файла. Строка Modes, являющаяся активной, зависит от глубины цвета; более подробно об этом см. раздел 22.3. Стандартные настройки X-сервера выбираются с учетом первого разрешения, указанного в активной строке Modes, однако вы можете переходить от одного разрешения к другому, удерживая нажатой клавиши <Alt> и <Control> и нажимая клавиши со знаком “плюс” или “минус” на цифровом блоке клавиатуре.

[4] При нажатии комбинации <Alt+функциональная клавиша> следует удерживать нажатой клавишу <Control>, чтобы X-сервер мог переключить виртуальные терминалы; для текстовой консоли нажимать клавишу <Control> не нужно.

При нажатии комбинации <Control+Alt+пробел> происходит немедленное уничтожение X-сервера. Если вы запустили сервер из консоли, вы окажетесь в ней после прекращения работы сервера. Если сервер был запущен диспетчером дисплеев, он запустит новый сервер и предложит снова ввести имя пользователя и пароль. Вам нужно будет уничтожить диспетчер дисплеев (`xdm`, `gdm` и т.п.) из текстовой консоли, чтобы он не порождал новые X-серверы.

Когда хороший X-сервер приходит в негодность

После того как вы вновь получите возможность управлять компьютером, вы сможете найти причину возникновения проблемы. Проще всего начать с анализа выходных данных X-сервера. Эти данные периодически появляются на первом виртуальном терминале (<Control+Alt+F1>), на который поступают все выходные данные программы запуска. Как правило, выходные данные X-сервера направляются в регистрационный файл (например, в файл `/var/log/Xorg.0.log`).

Как будет показано ниже, каждой строке предшествует символ, определяющий ее категорию. Вы можете использовать эти символы для обозначения ошибок (`EE`) и предупреждений (`WW`), а также для того, чтобы определить, как сервер будет находить каждую порцию информации: посредством стандартных настроек (==), в конфигурационном файле (**), путем автоматического обнаружения (--) или в командной строке X-сервера (++).

Давайте проанализируем следующий фрагмент кода.

```
X Window System Version 6.8.2
Release Date: 9 February 2005
X Protocol Version 11, Revision 0, Release 6.8.2
Build Operating System: Linux 2.4.21-23.ELsmp i686 [ELF]
Current Operating System: Linux chinook 2.6.12-1.1372_FC3 #1 Fri Jul 15 00:59:
   10 EDT 2005 i686
Markers: (--) probed, (**) from config file, (==) default setting,
     (++) from command line, (!!) notice, (II) informational,
     (WW) warning, (EE) error, (NI) not implemented, (??) unknown.
(==) Log file: "/var/log/Xorg.0.log", Time: Mon May 1 08:41:02 2006
(==) Using config file: "/etc/X11/xorg.conf"
(==) ServerLayout "Default Layout"
(**) |-->Screen "Screen0" (0)
(**) | |-->Monitor "Monitor0"
(**) | |-->Device "Videocard1"
(**) |-->Input Device "Mouse0"
(**) |-->Input Device "Keyboard0"
```

В первых строках указывается номер версии X-сервера и версия протокола X11, которую он реализует. В последующих строках сообщается о том, что сервер использует стандартные значения для местонахождения регистрационного файла, местонахождения конфигурационного файла, а также компоновки активного сервера. Дисплей и устройства ввода в конфигурационном файле отображаются в схематическом виде.

Одной из часто встречаемых проблем, которые можно увидеть в журнале, является сложность с определенными разрешениями экрана, о чем обычно свидетельствует сообщение об ошибке “Unable to validate any modes; failing back to the default mode” (Невозможно проверить какой-либо из режимов; производится возврат к стандартному режиму). Если не определить список частот для вашего монитора, X-сервер попытается использовать их с помощью Extended Display Identification Data (расширенные данные идентификации дисплея) (EDID). Если ваш монитор не поддерживает EDID, или если

ваш монитор выключен, когда запускается система X, вам нужно указать диапазон частот для системы X в секции `Monitor` конфигурационного файла.

Ошибка округления в результатах, полученных после тестирования EDID, может привести к тому, что некоторые разрешения будут недоступными, даже если они должны поддерживаться вашей видеоплатой и монитором. Об этом свидетельствуют такие записи в журнале, как "No valid modes for 1280x1024; removing" (Нет подходящих режимов для разрешения 1280x1024; удаление). Решить эту проблему можно, приказав X-серверу игнорировать информацию EDID и использовать частоты, определенные вами в следующих строках секции `Device`:

```
Option   "IgnoreEDID" "true"
Option   "UseEdidFreqs" "false"
```

В качестве другого примера предположим, что вы забыли правильно определить секцию мыши. В этом случае выходные данные должны содержать ошибку следующего рода:

```
(==) Using config file: "/etc/X11/xorg.conf"
Data incomplete in file /etc/X11/xorg.conf
  Undefined InputDevice "Mouse0" referenced by ServerLayout "Default Layout".
(EE) Problem parsing the config file
(EE) Error parsing the config file
Fatal server error:
no screens found
```

После того как система X заработает, и вы войдете в нее, вы сможете запустить команду `xdpyinfo`, чтобы получить дополнительную информацию о конфигурации X-сервера[5]. Данные, полученные после выполнения команды `xdpyinfo`, снова скажут вам об имени дисплея и предоставят информацию о версии X-сервера. В них будет сообщено также о доступной глубине цвета, загруженных расширениях и установленных экранах, их размерах и конфигурации.

Выходные данные команды `xdpyinfo` можно проверять синтаксически с помощью сценария (такого как `~/.xsession`), чтобы узнать размер активного экрана и правильно задать параметры рабочего стола.

В процессе отладки команду `xdpyinfo` полезно выполнять для того, чтобы определить, что X-сервер работает и следит за запросами в сети, что у него правильно сконфигурирован экран и разрешение, и что он работает с требуемой глубиной цвета. Если все это подтверждается, можно запускать X-приложения.

22.5. Краткое замечание о настольных средах

Гибкость и простота модели клиент/сервер в системе X с годами привели к появлению наборов библиотек графических интерфейсных элементов, диспетчеров окон, обозревателей файлов, утилит панели инструментов и полезных программ. Группа Project Athena (из Массачусетсского технологического института), которая, собственно, и разработала систему X, создала библиотеки графических интерфейсных элементов Athena и утилиту `twm` (расшифровывается как Tom's Windows Manager — диспетчер окон Тома, названный в честь ее создателя Тома Ластранжа (Tom LaStrange); его еще называют Tab

[5] Входить в систему X от имени суперпользователя не рекомендуется, поскольку эта операция может создать множество стандартных файлов запуска в домашнем каталоге суперпользователя, которым обычно является каталог `/`. Кроме того, это небезопасно. Наоборот, входить в систему нужно от имени обычного пользователя и использовать утилиту `sudo`. В системах Debian и Ubuntu эта схема принята по умолчанию.

Window Manager). Эти элементарные инструменты являлись фактическим стандартом первых X-приложений.

OpenLook, разработанный фирмами Sun Microsystems и AT&T, был альтернативным инструментарием, в котором для отображения меню и диалоговых окон использовались овальные и канцелярские кнопки. Примерно в то же время Open Software Foundation представила конкурирующую платформу Motif (позже получившую название CDE, или Common Desktop Environment — общая среда рабочего стола), которая в итоге была одобрена фирмой Sun. В течение некоторого времени трехмерные изображения в этих инструментариях являлись элегантными, а широкое распространение усовершенствованных элементов графического интерфейса, таких как ползунки и меню, сделали их разумным выбором для новых программ. Однако оба инструментария были слишком частными, а условия лицензирования, распространяемые на библиотеки разработки и диспетчер окон, сделали их недоступными для широкой публики.

Наряду с операционными системами с открытым исходным кодом, такими как Linux, стали появляться и среды рабочего стола, тоже с открытым исходным кодом. FVWM ("F" Virtual Window Manager — виртуальный диспетчер окон "F") был популярным для Linux из-за широких возможностей его конфигурирования и поддержки "виртуальных рабочих столов", которые позволили расширить эффективную рабочую область пользователя за рамки дисплеев с низким разрешением, доступных на большинстве персональных компьютеров[6]. Однако тогда не существовало связанных с ними библиотек графических интерфейсных элементов, поэтому пользователи были вынуждены работать с множеством программ, каждая из которых обладала отличным внешним видом.

По мере того как приложения становились более совершенными, и для них требовалось все больше продвинутых функций пользовательского интерфейса, стало ясно, что для объединения опыта пользователей и предоставления более качественной поддержки разработчикам приложений нужен был более широкий подход. Как результат, в современных средах рабочего стола для Linux мы имеем двух крупных игроков: GNOME и KDE. Несмотря на то что некоторые пользователи имеют свое понимание того, какая из них является самым "истинным путем", обе среды являются полноценными диспетчерами рабочего стола. В сущности, работая в одном "королевстве", вы можете использовать приложения из другого "королевства". Вы должны лишь быть готовы к тому, что приложение будет выглядеть и вести себя по-другому.

Проект freedesktop.org посвящен созданию среды, которая позволит приложениям быть совместимыми с любой настольной средой.

KDE

Среда KDE (расшифровывается как "K Desktop Environment" — настольная среда K) написана на языке C++ и основана на библиотеке инструментов Qt. Ее предпочитают использовать те пользователи, которым нравятся привлекательные элементы и эффекты интерфейса (прозрачные окна, тени и анимированные указатели мыши). Выглядит она привлекательно, однако она может замедлить работу малопроизводительных персональных компьютеров. Работать в этой среде будет удобно тем пользователям, которые большую часть времени тратят на щелчки на рабочем столе, а не на запуск приложений.

Среду KDE часто предпочитают пользователи, перешедшие из Windows или Mac, и причина тому — все та же привлекательность графики. Она придется по вкусу также тем поль-

[6] В действительности FVWM был настолько гибким, что его можно было сконфигурировать так, чтобы он был похож или на *twm*, или на *mwm* (Motif Window Manager).

зователям, кому нравится самостоятельно настраивать все параметры среды. Для остальных пользователей среда KDE вряд ли подойдет — они могут воспользоваться средой GNOME.

Приложения, написанные для среды KDE, практически всегда содержат букву "K" в своем названии (например, Konqueror — обозреватель Web или файлов, Konsole — имитатор терминала, KWord — текстовый процессор). Стандартный диспетчер окон, KWin, поддерживает стандарт freedesktop.org Window Manager Specification, конфигурируемые оболочки (skin) для изменения внешнего вида, а также многие другие особенности. Комплект приложений KOffice позволяет обрабатывать тексты, электронные таблицы и проводить презентации. KDE предлагает обширный набор инструментов для разработки, включая интегрированную среду разработки (IDE). Благодаря всем этим особенностям, KDE предстает как мощный пользовательский интерфейс.

GNOME

Среда GNOME — это другая настольная среда, написанная на языке C, и основанная на библиотеке графических интерфейсных элементов GTK+. Базовая модель объектных соединений этой среды использует технологию CORBA (Common Object Request Broker Architecture — Общая архитектура посредника запросов к объектам). Название GNOME первоначально являлось акронимом GNU Network Object Model Environment (Среда сетевых объектных моделей GNU), однако это происхождение в наши дни не имеет никакого значения — сейчас существует просто имя GNOME.

Среда GNOME менее броская, чем KDE, предлагает меньше возможностей для конфигурирования, и вообще является менее согласованной. Однако она гораздо понятнее, быстрее и проще, чем KDE. Среда GNOME используется во многих дистрибутивах в качестве стандартной настольной среды. Подобно KDE, GNOME имеет богатый набор приложений. Ее приложения обычно идентифицируются буквой "G" в названии приложения. Исключением является стандартный диспетчер окон, называемый Metacity, который предлагает базовые функции работы с окнами и оболочки, позволяющие изменять внешний вид. В соответствии с моделью GNOME, Metacity обязана быть "шустрой и проворной". Если вам понадобятся какие-то дополнительные особенности вроде виртуального рабочего стола или интеллектуального размещения окон, вам нужна будет поддержка внешних приложений, таких как `brightside` или `devilspie`. (Это единственная область, в которой KDE занимает неплохие позиции.)

К офисным приложениям относятся AbiWord для обработки текстов, Gnumeric для работы с электронными таблицами, а также один из наиболее впечатляющих проектов — GIMP, позволяющий производить обработку изображений. Еще есть диспетчер файлов Nautilus и Web-браузер Epiphany. Как и KDE, GNOME предлагает обширную инфраструктуру для разработчиков приложений. Подводя итоги, можно с уверенностью сказать, что GNOME предлагает функциональную архитектуру для разработки приложений в простой в использовании среде рабочего стола.

Что лучше: GNOME или KDE?

Попробуйте-ка задать такой вопрос на одном из открытых форумов, как тут же начнется "крупная война". Вопрос о предпочтениях практически всегда затрагивает личные чувства и становится причиной настоящих "крестовых походов", поэтому вы можете не согласиться с тем, что написано в следующих двух абзацах.

Конечно же, лучше всего попробовать в деле обе среды и решить самостоятельно, какая из них подходит вам больше всего. Следует иметь в виду, что ваши приятели, ваши

пользователи и ваш администратор могут иметь совершенно разные предпочтения относительно среды рабочего стола, и в этом нет ничего плохого.

Сейчас, когда проект freedesktop.org занимается выработкой стандартов для объединения рабочего стола, враждебность, возникшая между лагерями приверженцев KDE и GNOME, постепенно перерастает в здоровую конкуренцию, что побуждает разрабатывать более качественное программное обеспечение. Не забывайте, что ваш выбор, сделанный в пользу той или иной среды, не означает, что вы сможете работать только с некоторыми приложениями. Не важно, какую среду вы выберете — вы сможете выбирать приложения из всего имеющегося программного обеспечения, доступного для каждого из этих (и других) проектов с открытым исходным кодом.

22.6. Рекомендуемая литература

- На домашней странице X.Org (`www.x.org`) содержится информация о свежих выпусках, а также ссылки на проект Wiki X.Org, списки рассылки и ссылки для загрузки.
- На `man`-страницах `Xserver` и `Xorg` рассмотрены общие опции X-сервера и специфические опции командной строки `Xorg`. Они содержат также обзор принципов работы X-сервера. На `man`-странице `xorg.conf` приводится конфигурационный файл с подробным описанием его секций. На этой странице также перечислены драйверы видеоплат в разделе `REFERENCES` (Справка). Проверьте на этой странице, какой драйвер управляет вашей видеоплатой, а затем ознакомьтесь с `man`-страницей, чтобы узнать обо всех опциях этого драйвера.

22.7. Упражнения

22.1. Используя SSH, запустите программу X по сети. Используйте команду `ssh -v`, чтобы удостовериться, что перенаправление настроено должным образом. Какое значение будет иметь переменная окружения `DISPLAY` после того, как вы войдете в систему? Выведите на экран список cookie-наборов, выполнив команду `xauth`, и убедитесь в том, что для данного дисплея включена проверка достоверности с использованием cookie-наборов.

22.2. Напишите команду интерпретатора команд или сценарий для проверки синтаксиса выходных данных `xdpyinfo`, и напечатайте текущее разрешение экрана в формате X×Y (например, 1024×768).

22.3. Проверьте регистрационный файл `Xorg` (`/var/logXorg.o.log`), и на его основе постарайтесь ответить на следующие вопросы:

а) Какая видеоплата используется, и какой драйвер управляет ее работой?

б) Какой объем памяти установлен на видеоплате?

в) Использовались ли данные EDID для тестирования настроек монитора? Как это можно узнать?

г) Какие режимы (разрешения) поддерживаются?

д) Включена ли функция DPMS?

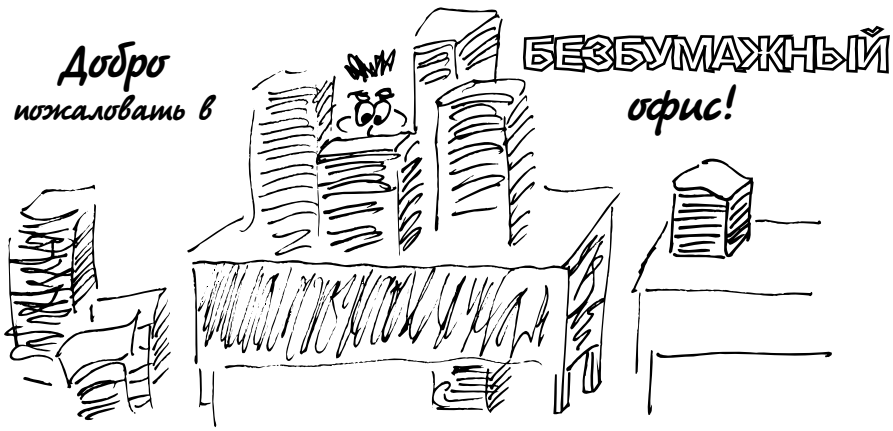
е) О каких физических размерах экрана известно серверу?

ж) Какой файл устройства используется для мыши?

22.4. Какой флаг отключает нелокальные TCP-соединения с сервером? Объясните, чем полезна эта опция?

Глава 23

Печать



Конфигурирование принтера — это утомительная и сложная задача. Пользователи воспринимают печать как должное, но в действительности для того, чтобы они получали качественный результат при печати на своем принтере, администратору может потребоваться приложить немало усилий.

Двадцать лет назад самыми распространенными считались построчно печатающие ASCII-принтеры. Лазерные принтеры стоили дорого и встречались редко. Для устройств с высокой разрешающей способностью требовались специальные драйверы и программы форматирования.

Сегодня лазерные принтеры уже часто подключаются не к одному компьютеру через последовательный или параллельный порт, а к сети TCP/IP через Ethernet или беспроводное соединение. Лазерные принтеры по большей части уступили нижний сегмент рынка струйным принтерам. Цветные принтеры когда-то были роскошью, но подобно цветной фотографии и цветным мониторам тоже стали обычным явлением. Отыскать черно-белый принтер скоро будет также трудно, как найти черно-белую пленку.

На рынке настольных систем и небольших офисов, специализированные принтеры, сканнеры, ксероксы и факсы начинают сменять многофункциональные устройства, которые умеют выполнять сразу все эти задачи. Иногда эти устройства умеют даже считывать файлы с карты памяти цифровой камеры.

С учетом всех этих многочисленных изменений в технологии можно предположить, что система печати в Linux является гибкой. И это действительно так. Однако подобная гибкость была достигнута относительно недавно. Еще несколько лет назад почти все системы печати в Linux основывались на программном обеспечении, которое разрабатывалось еще для построчно печатающих принтеров. И хотя эти системы модернизирова-

лись и расширялись так, чтобы "идти в ногу" с появляющимися новыми технологиями, они никогда не смогли бы поддерживать современные принтеры в современных сетях. К счастью, была разработана новая система CUPS (Common UNIX Printing System — Общая система печати в UNIX), решающая многие из проблем старых систем.

Сегодня CUPS используется на почти всех современных системах UNIX и Linux, включая Mac OS X. Из более старых систем печати до сих пор встречаются лишь несколько (типа PDQ, `pdq.sourceforge.net`, и LPRng, `www.lprng.com`), а первые системы печати, подобные системам печати System V, Palladium, rlpr, PLP, GNUlpr и PPR, уже почти совсем исчезли.

В этой главе основное внимание будет уделяться системе CUPS, ведь именно она считается текущим стандартом. Начнем мы с общего знакомства с принтерами и связанной с ними терминологией, затем рассмотрим существующие в Linux системы печати в общем и архитектуру системы CUPS — в частности. Далее мы расскажем об особенностях конфигурирования (настройки) и администрирования принтеров и, наконец, напоследок, предоставим несколько рекомендаций по отладке систем печати, приведем перечень дополнительного программного обеспечения для печати и дадим несколько общих советов касательно администрирования.

23.1. Принтеры становятся более сложными

Пользователи используют принтеры вместе с другими периферийными устройствами, такими как мониторы и колонки, но сложными принтеры называют не за это. Когда-то самым мощным считался компьютер Apple LaserWriter. Сегодня настольный компьютер возможно и является мощнее принтера, но принтер — это все равно что компьютер. У него имеется ЦП, память, операционная система и возможно даже диск. Если это сетевой принтер, у него имеется свой собственный IP-адрес и реализация TCP/IP.

Если у вас есть под рукой современный сетевой принтер, попробуйте ввести его сетевой адрес в окне своего браузера (например, 192.168.0.9). Наверняка принтер вернет какие-нибудь Web-страницы, позволяющие администрировать его оборудование; это означает, у принтера может быть свой собственный Web-сервер.

Поскольку системных администраторов всегда волнует вопрос безопасности, возможно, вы уже подумали, а не означает ли это также, что принтер может подвергаться несанкционированному доступу и атакам типа отказа от обслуживания. Конечно, может! Подробнее об этом будет рассказываться далее в этой главе, в разделе, посвященном безопасности.

Под управлением какой операционной системы работает ваш принтер? Что? Вы не знаете? Не удивительно. И возможно вам и не удастся узнать это, даже покопавшись в соответствующей документации. Используемая принтером операционная система зависит от производителя, а иногда даже и от модели. Принтеры средней и высокой мощности могут даже использовать какую-нибудь производную версию UNIX или Linux[1].

Неразбериха с операционными системами — только начало. Принтеры также поддерживают разные сетевые протоколы и принимают задания на нескольких разных, являющихся специфическими для каждого принтера языках описания страниц и документов.

Если вы работаете администратором на большом предприятии, вам может понадобиться поддерживать несколько моделей принтеров от нескольких разных производи-

[1] Хакерам удалось перенести Linux на iPod и Xbox; теперь ждем, кому первому удастся перенести ее на HP LaserJet.

телей. Программные средства печати на ваших компьютерах должны быть готовы взаимодействовать с разными (и иногда даже неизвестными) аппаратными средствами и использовать ряд протоколов.

23.2. Языки принтеров

Задание на печать — это на самом деле компьютерная программа, написанная на специальном языке программирования. Такие языки вместе называются языками описания страниц или просто PDL-языками (Page Description Language — язык описания страниц).

Страницы, кодируемые на языке PDL, могут быть гораздо меньше по размеру и передаваться гораздо быстрее аналогичных необработанных изображений. Кроме того, PDL-описания также могут не зависеть ни от используемых устройств, ни от их разрешающей способности.

Наиболее известными PDL-языками на сегодняшний день являются PostScript, PCL5, PCL6 (также называемый "PCL/XL" или "pxl") и PDF. Многие принтеры способны принимать входные данные на более чем одном языке. Каждый из этих языков коротко описывается в следующих ниже разделах.

Принтеры должны интерпретировать задания на этих языках и преобразовывать их в какой-нибудь понятный для используемого устройства обработки изображений формат. Поэтому принтеры содержат языковые интерпретаторы. Точно так же как и у языка C или Java, у этих языков имеется множество версий, и каждая из этих версий работает по-своему. Почти все принтеры PostScript понимают PostScript Level 3, но если вы отправите программу Level 3 на принтер, который понимает только Level 2, принтер, скорее всего, не сможет ее выполнить. Разве стали бы вы компилировать программу на языке FORTRAN 90 компилятором FORTRAN 77? Разумеется, нет.

Процесс преобразования PDL-описания (или чего-нибудь еще типа файлов изображений) в растровые изображения страниц называется *обработкой растровых изображений*, а программа, которая выполняет его — *процессором растровых изображений* (Raster Image Processor), или просто *RIP*.

Задания на печать можно копировать на компьютер и просматривать на экране. Централизованные интерпретаторы, которые позволяют это делать, такие как Ghostscript, описываются в разделе 23.7. В теории можно даже копировать задания на компьютер и затем отправлять готовые (и значительно большие по размеру) растровые изображения на печать какому-нибудь "не очень интеллектуальному" устройству печати. На самом деле именно таким образом и работают многие принтеры GDI (Windows), да и в Linux этот подход тоже поддерживается до некоторой степени.

PostScript

PostScript является самым распространенным из всех встречающихся на системах Linux языков семейства PDL. Первоначально он был разработан компанией Adobe Systems, и многие принтеры до сих пор используют интерпретатор, требующий наличия лицензии от Adobe. Почти все программы компоновки страниц умеют генерировать данные на языке PostScript, а некоторые вообще работают только с такими данными.

PostScript — этот полнофункциональный язык программирования. Почти все написанные на нем программы могут просматриваться с помощью текстового редактора или утилиты `less`. Эти программы отличаются обилием круглых и фигурных скобок, а

также символов косой черты и часто начинаются символами `%!PS`. Хотя эти начальные символы не являются обязательными в самом языке PostScript, PostScript-интерпретаторы и другие программы для печати часто ищут их, когда пытаются распознать и классифицировать задания на печать.

PCL

Одной из альтернатив языка PostScript является разработанный компанией Hewlett-Packard язык PCL (Printer Control Language — язык управления принтерами). Его понимают не только принтеры производства Hewlett-Packard, но и принтеры многих других производителей; некоторые принтеры вообще умеют работать только с ним.

В отличие от PostScript, который соответствует концепции машины Тьюринга и является универсальным языком программирования, PCL просто сообщает принтерам, как им следует печатать страницы. Написанные на PCL задания на печать имеют бинарный, не доступный для человеческого восприятия формат и обычно намного короче аналогичных заданий, написанных на языке PostScript. Приложения Linux редко генерируют выходные данные на языке PCL напрямую, но фильтры могут преобразовывать PostScript в PCL.

В отличие от языка PostScript, версии языка PCL разнятся лишь немного. Отличия являются небольшими, но достаточно существенными для того, чтобы вызывать раздражение. Задания, которые печатаются правильно на LaserJet 5si, могут печататься немного не так на LaserJet 5500, и наоборот. И это касается не только этой пары моделей; каждый PCL-принтер использует свой диалект языка PCL со своими специальными командами, которые извлекают пользу из функциональных возможностей именно этого принтера.

Например, если сказать компьютеру, что используется принтер LaserJet 4500, когда на самом деле используется принтер LaserJet 4550, он может сгенерировать какие-нибудь PCL-команды, которые 4550 проигнорирует или интерпретирует неправильно. Также если имеется какое-нибудь сохраненное, написанное на языке PCL задание на печать — скажем, чистый бланк на оформление покупки — а принтер, для которого оно было сгенерировано, заменяется каким-то более новым, не исключено, что это задание придется генерировать заново.

Даже еще хуже то, что компания HP (Hewlett-Packard) определила два совершенно не связанных между собой семейства языков — PCL: PCL5 (5C означает “цветной”, 5E — “черно-белый”) и PCL6 (также называемый “PCL/XL”). Сегодня новые принтеры HP обычно имеют языковые интерпретаторы для них обоих.

PCL4 — это архаичная разновидность PCL5. К принтеру PCL4 (и еще старше) следует относиться так же, как и к интерпретатору Perl 4: лучше заменить его чем-то более новым.

PDF

Разработанный компанией Adobe формат PDF (Portable Document Format — формат переносимых документов) генерируется программой Adobe Acrobat, а многие другие настольные средства для издательского дела, такие как OpenOffice, позволяют экспортировать документы в этом формате.

PDF-документы не зависят от платформы. Формат PDF очень часто используется для обмена электронными документами с возможностью последующего просмотра и пе-

чати. Окончательный вариант текста данной книги, например, передавался на принтер именно в виде PDF-файла.

PDF — это язык описания документов, а не просто язык описания страниц. Он позволяет описывать не только отдельные страницы, но и также общую структуру документа: к каким главам относятся те или иные страницы, какие текстовые столбцы перетекают в другие текстовые столбцы и т.д. Он также предлагает целый ряд различных мультимедийных возможностей для использования на экране.

Некоторые принтеры интерпретируют PDF напрямую. Если ваш этого не делает, существует масса программ-трансляторов и программ для просмотра PDF-файлов (среди них Ghostview, `xpdf`, `kpdf`, Evince и Acrobat Reader), которые позволяют преобразовывать документы в какой-нибудь более универсальный формат (такой как PostScript, например). Ваша система печати может даже скрывать требование на выполнение преобразования и автоматически преобразовывать PDF-документы перед их отправкой на принтер.

XHTML

Язык XHTML-Print появился недавно. Принтер, который получает поток данных на языке XHTML-Print (описывающих, к примеру, Web-страницу) генерирует качественное представление задания, но разные принтеры могут генерировать разные представления, точно так же как разные браузеры могут по-разному представлять одну и ту же Web-страницу.

Зачем это может быть нужно пользователям? Предположим, что вы — начальник отдела маркетинга и что, просматривая Web-страницы на своем телефоне, вы вдруг обнаруживаете среди них одну, очень подходящую к теме вашей презентации, которую вы как раз собираетесь провести. Вы подходите к ближайшему поддерживающему Bluetooth принтеру и отправляете ему со своего телефона URL-адрес этой страницы. Дальше принтер делает все сам: загружает страницу из Internet, визуализирует ее и печатает копии. Вы берете копии из лотка и спокойно направляетесь на свою презентацию.

PJL

PJL (Printer Job Language — язык заданий принтеров; был разработан компанией Hewlett-Packard) — это не PDL-язык, а мета-язык, который описывает задания принтеров. Мы рассказываем о нем здесь потому, что он будет встречаться в описаниях принтеров.

Итак, PJL — это язык управления заданиями, который позволяет указывать вещи, подобные тому, какой PDL-язык должен использоваться для задания, дуплексное это задание или симплексное, какой размер бумаги следует использовать, и т.д. PJL-команды находятся в начале задания, а PJL-операторы все начинаются с символов `@PJL`:

```
@PJL SET COPIES=3
@PJL COMMENT FOO BAR MUMBLE
@PJL SET DUPLEX=ON
@PJL SET PAGEPROTECT=OFF
@PJL ENTER LANGUAGE=PCL
```

PJL обычно нормально распознается (или преднамеренно игнорируется) принтерами производства не Hewlett-Packard, но в случае возникновения проблем при распечатке содержащего PJI документа на таком принтере попробуйте удалить PJL при помощи текстового редактора и снова отправить задание на печать.

Драйверы принтеров и поддержка ими PDL-языков

Программное обеспечение, которое преобразовывает файл в понятный для данного принтера формат, называется *драйвером принтера*. Чтобы печатать на принтере LaserJet 5500 данные в формате PCL5, необходим драйвер LaserJet 5500 PCL5.

А что если принтер поддерживает только некоторые из нужных языков? Что делать, если вы загрузили из Internet файл PostScript, а имеющийся принтер распознает только PCL5E? Как распечатать PDF-файл, если принтер не интерпретирует PDF напрямую?

Один из вариантов — преобразовать файл вручную. Дистрибутивы Linux поставляются с множеством утилит преобразования; практически всегда есть какой-нибудь способ превратить имеющийся документ в документ, который может быть распечатан на принтере. Браузеры, например, позволяют преобразовывать страницы HTML (или XHTML) в PostScript. OpenOffice позволяет преобразовывать файлы MS Word в формат PDF. Ghostscript позволяет преобразовывать PDF-файлы в формат PostScript, а уже из формата PostScript — в почти любой другой формат, включая PCL.

Более простой вариант — это позволить системе печати сделать все самостоятельно. Многие системы имеют встроенные знания о том, какие преобразования должны выполняться, и могут выполнять их автоматически.

Если необходимо определить, какой PDL-язык используется в файле, и сделать это по имени файла (например, `food.pdf`) не удается, можно воспользоваться командой `file` (если только файл не начинается с ряда PJL-инструкций, потому что в таком случае команда `file` вернет просто сообщение “HP Printer Job Language data” (данные на языке заданий принтеров HP)).

Сохраните несколько заданий на печать в файлах вместо того, чтобы отправлять их на принтер, и сможете увидеть, как выглядит программа на одном из этих языков. Поизучав файлы каждого из этих типов минуту или две в своем текстовом редакторе, вы поймете, насколько разными они являются. Не выводите их прямо на экран командой `cat`, потому что только PostScript отображается в формате ASCII.

PostScript:

```
%!PS-Adobe-3.0
%%BoundingBox: 0 0 612 792
%%Pages: 1
% ...
% Draw a line around the polygons... (Нарисовать линию вокруг многоугольников)
pop pop pop dup 0 setgray 0 0 moveto dup 0 lineto 0.707106781 mul dup
   lineto closepath stroke
```

PDF:

```
%PDF-1.3
%A.A.AA"
81 0 obj
<<
/Linearized 1
/O 83
/H [ 915 494 ]
/T 125075
>>
endobj
xref
81 24
```

```
0000000016 00000 n
A^<8f>
^P^@A¤A'<9e>
endstream
endobj
```

PCL5:

```
^[E^[&l1o0o1t0l6D^[&l1X^[*r0F^[*v0n1O^[*p4300X^[%1BDT~,1TR0TD1SP1FT10,50
CF3,1LB.~;^[%1A^[*c100G^[*v2T^[&a0P^[*p0X^[*p0Y^[(10U^[(s1p12vsb4148T^[&l0
E^[*p0Y^[*ct7920Y^[(10U^[(s1p12vsb4101T^[&a0P^[&l0o66f0E^[9^[&a0P^[*p0X^[*
p0Y^[*p474Y^[*p141X^[(10U^[(10U^[(s1p12vsb4101T^[*p402Y^[*p186X^[*v0O^[*c9
00a4b100g2P^[*v1O^[*p250Y^[*v0O^[*c900a4b100g2P^[*v1O^[*v0O^[*c4a156b100g2
P^[*v1O^[*p251Y^[*p187X^[*v0O^[*c899a154b10g2P^[*v1O^[*p346Y^[*p256X
```

PCL/XL:

```
A'X^BX^BA.<89>A^@A.<86>A^CA.<8f>AA^@A.<88>A^AA.<82>HA^@A.(A^@A.%A
A.cA^A^P^@TimesNewRmnBdA.A.A…UUA©BA.A¦Au^BA.A.o<85>A‟A›^CA^BA.Lk
Af^@^@A.A.dA^A:^@
```

23.3. Архитектура CUPS

CUPS имеет клиент-серверную архитектуру. Сервер CUPS — это спулер, который обслуживает очереди клиентских заданий на печать.

Клиентами CUPS могут быть как приложения с интерфейсом командной строки (вроде `lpr` и `lpq`), так и приложения с графическим пользовательским интерфейсом (наподобие `kprinter`). Другие серверы CUPS тоже могут выступать в роли клиентов с точки зрения данного сервера.

Давайте возьмем самую простую из всех возможных конфигурацию: сервер CUPS на одной машине, подключенной к одному принтеру, с одной очередью заданий на печать. В следующих нескольких разделах дается краткий обзор команд и процессов, используемых при выполнении нескольких наиболее распространенных, связанных с печатью задач.

Печать документов

Вот как можно было бы распечатать файлы `foo.pdf` и `/tmp/testprint.pdf`:

```
$ lpr foo.pdf /tmp/testprint.ps
```

Клиент передает копии файлов серверу CUPS, который сохраняет их в очереди на печать. Когда принтер готов, CUPS начинает обрабатывать каждый файл по очереди.

CUPS проверяет как документ, так и PPD-файл принтера (PostScript Printer Description — описание принтеров в PostScript), дабы узнать, что необходимо сделать для того, чтобы документ был распечатан должным образом. (Как будет объясняться позже, PPD-файлы используются даже и для не-PostScript принтеров.)

Чтобы подготовить задание к печати на конкретном принтере, CUPS пропускает его через конвейер фильтров. Эти фильтры могут выполнять самые разные функции. Например, фильтр может изменять формат задания так, чтобы на каждой физической странице печаталось по два уменьшенных в размере изображения страниц, или преобразовывать задание из одного PDL в другой. Фильтр также может выполнять какую-нибудь специфическую для принтера операцию обработки, такую как инициализация

принтера. Фильтр даже может преобразовывать изображения в растровый формат вместо принтеров, если у тех нет своего собственного процессора RIP.

Последним этапом в конвейере печати является серверный компонент, который отправляет задание с хоста на принтер через подходящий протокол, такой как USB. Он еще также отправляет информацию о состоянии обратно на сервер CUPS. Просмотреть доступные серверные компоненты можно при помощи такой команды:

```
$ locate backend | grep -i cups
```

Передав задание на печать, демон CUPS возвращается снова к обработке своих очередей и запросов от клиентов. А принтер приступает к работе, пытаясь распечатать переданное ему задание.

Просмотр и манипулирование очередями на печать

Команда `lpq` запрашивает с сервера CUPS информацию о состоянии задания и форматирует ее для отображения.

Команда `lpstat -t` предоставляет хороший сводный отчет об общем состоянии сервера печати.

Клиенты CUPS могут просить сервер откладывать задания, отменять их или изменять их приоритет. Они также могут переносить задания из одной очереди в другую. Почти все изменения требуют указания номера задания, который сообщается в отчете, возвращаемом командой `lpq`.

Например, чтобы удалить какое-нибудь задание, нужно просто выполнить следующую команду:

```
lprm идентификатор_задания
```

Множество принтеров

Если к машине подключено несколько принтеров, CUPS создает для каждого из них отдельную очередь.

Клиенты командной строки принимают опцию (обычно `-P` *принтер* или `-p` *принтер*) для указания очереди принтера. Также можно самостоятельно задать принтер, который должен использоваться по умолчанию, просто установив переменную окружения `PRINTER`:

```
$ export PRINTER=имя_принтера
```

или указав CUPS использовать определенный параметр по умолчанию для данной учетной записи:

```
$ lpoptions -dимя_принтера
```

`lpoptions` обычно устанавливает личные параметры по умолчанию пользователя, которые хранятся в каталоге `~/lpoptions`. При выполнении от имени пользователя root она устанавливает системные параметры по умолчанию, которые хранятся в каталоге `/etc/cups/lpoptions`. `lpoptions -l` отображает список текущих опций.

Экземпляры принтеров

Когда имеется только один принтер и его нужно использовать по-разному — например, как для быстрой печати черновых вариантов, так и для качественной печати про-

изводственных документов — CUPS позволяет создавать для выполнения этих разных задач разные "экземпляры принтера".

Например, если уже есть принтер с именем Phaser_6120, команда

```
$ lpoptions -p Phaser_6120/2up -o number-up=2 -o job-sheets=standard
```

создаст экземпляр с именем Phaser_6120/2up, печатающий по две страницы на листе и добавляющий титульные страницы. А команда

```
$ lpr -P Phaser_6120/2up biglisting.ps
```

затем распечатает PostScript-файл `biglisting.ps` как задание с двумя страницами на каждом листе и соответствующими титульными страницами.

Сетевая печать

С точки зрения системы CUPS сеть с множеством машин не очень отличается от изолированной машины. На каждом компьютере выполняется демон CUPS (`cupsd`), и все эти демоны взаимодействуют между собой.

Сконфигурировать демон CUPS так, чтобы он принимал задания на печать с удаленных систем, можно отредактировав файл `/etc/cups/cupsd.conf` (подробнее об этом будет рассказываться чуть позже в этой главе, в разделе 23.4). Настроенные таким образом серверы CUPS по умолчанию каждые 30 секунд рассылают информацию об обслуживаемых ими принтерах. Благодаря этому компьютеры локальной сети автоматически узнают о доступных им принтерах.

Сделать принтеры доступными для множества сетей или подсетей немного сложнее, потому что рассылаемые пакеты не пересекают границ подсетей. Наиболее популярным решением является выделить в каждой подсети подчиненный сервер, который будет запрашивать информацию с серверов в других подсетях и затем рассылать ее машинам в своей локальной подсети.

Например, предположим, что сервер allie (192.168.1.5) и сервер jj (192.168.2.14) находятся в разных подсетях, и что необходимо, чтобы оба из них были доступны пользователям в третьей подсети, 192.168.3. Чтобы решить эту проблему, нужно просто выделить подчиненный сервер (например, copeland, 192.168.3.10) и добавить в его файл `cupsd.conf` следующие строки:

```
BrowsePoll allie
BrowsePoll jj
BrowseRelay 127.0.0.1 192.168.3.255
```

Первые две строки указывают демону `cupsd` подчиненного сервера запрашивать у находящихся на сервере allie и jj демонов `cupsd` информацию об обслуживаемых ими принтерах. Третья строка указывает copeland рассылать получаемую информацию по своей собственной подсети.

Необходима более тонкая настройка? Множество очередей для одного принтера и чтобы у каждой были свои параметры по умолчанию? Один сервер, балансирующий нагрузку путем распределения заданий между несколькими принтерами? Множество серверов, способных работать с равнозначными экземплярами одного и того же принтера? Клиенты LPD или Windows? Вариантов слишком много, чтобы их все можно было рассмотреть здесь, но CUPS имеет решение для всех этих ситуаций, и узнать о них подробнее можно в документации CUPS.

Лежащий в основе CUPS протокол: HTTP

HTTP — это протокол, лежащий в основе всех взаимодействий между серверами CUPS и клиентами. Серверы CUPS для обнаружения подключений прослушивают порт 631. Клиенты отправляют задания при помощи операции HTTP POST. Запросы информации о состоянии реализуются посредством операции HTTP GET. Конфигурационные файлы CUPS по своему внешнему виду очень напоминают конфигурационные файлы Apache.

Краткий экскурс в историю позволит понять, как так получилось.

Первым коммерческим программным обеспечением в UNIX было программное обеспечение для производства документов. Главным образом это были текстовые редакторы, языки разметки (`nroff/troff`) и программы для печати.

Принтеры были примитивными, и такими же были и спулеры. С не-UNIX системами дело обстояло точно также, хотя не UNIX-системы были специализированными: системы IBM знали, как управлять принтерами IBM, компьютеры Apple знали, как управлять принтерами Apple, и т.д. Используемый компьютер часто должен был (по правилам) подключаться к принтеру напрямую. Настройка принтера заключалась в предоставлении ответов на вопросы типа “Последовательный или параллельный интерфейс?”.

Когда стали доступны сетевые принтеры, количество проблем только возросло. Первые сетевые системы печати были слишком “уникальными” и использовали целый ряд протоколов для обеспечения связи между принтером и спулером, между клиентом и спулером и для осуществления обмена сетевым трафиком.

Поскольку технологии становились все сложнее и сложнее, предпринимались попытки создать универсальный стандарт, но не одна из них не имела должного успеха. Используемые протоколы устаревали и начинали работать всех хуже и хуже. Новые принтерные возможности, такие как дуплексный обмен, тоже порождали множество проблем с хакерством.

Стиснув зубы, рабочая группа по решению проблем печати (Printing Working Group) из IEFT создала протокол IPP (Internet Printing Protocol — протокол печати через Internet), строящийся поверх протокола HTTP. Он не только оформлял взаимодействия в виде простых запросов GET и POST, но и также позволял системам печати пользоваться преимуществами стандартных, широко используемых технологий для аутентификации, контроля доступа и шифрования.

Майкл Свит (Michael Sweet) и Эндрю Сенфт (Andrew Senft) из компании ESP (Easy Software Products) перенесли IPP в UNIX в виде реализации CUPS. Сегодня CUPS является самой полноценной реализацией IPP на планете.

Хотя компания EPS имеет свой собственный сегмент на рынке и массу коммерческих продуктов, CUPS представляет собой ее проект с открытым исходным кодом, который может распространяться абсолютно бесплатно. В почти всех системах Linux и UNIX сегодня CUPS используется в качестве системы печати по умолчанию.

Сервер CUPS — это на самом деле Web-сервер, но только для взаимодействия он использует порт 631, а не 80. Убедиться в этом можно, подключившись к своему локальному серверу CUPS через Web-браузер (`localhost:631`) и увидев, что у сервера CUPS имеется свой собственный графический интерфейс пользователя, в котором отображаются все его функциональные возможности. (Для безопасного взаимодействия с принтерами также можно использовать SSL и порт 433). CUPS использует IPP для соединения с Web-браузерами, принтерами, приложениями с графическими пользовательским интерфейсом (GUI) и с интерфейсом командой строки и другими серверами CUPS.

PPD-файлы

При вызове `kprinter` для распечатки `book.ps` на цветном принтере Pollux программа `kprinter` может вернуться и спросить, бумага какого размера должна использоваться для печати. Но подождите, откуда CUPS знает, что ей нужно указать своему клиенту, `kprinter`, что Pollux может выполнять печать на бумаге размера A4? Откуда CUPS известно, что Pollux может обрабатывать PostScript, и что ей делать, если это не так? Где CUPS находит информацию о том, что Pollux является цветным принтером?

Вся эта информация содержится в файле PPD (PostScript Printer Description — файл описания принтера в PostScript), который описывает атрибуты и возможности принтера PostScript. Демон CUPS считывает PPD-файлы обслуживаемых им принтеров и передает информацию о них клиентам и, если необходимо, фильтрам.

PPD-файлы изначально разрабатывались для мира Mac, но были быстро переняты программами Windows. Каждый новый принтер поставляется с PPD-файлом от производителя. Драйверы принтеров Mac и Windows используют PPD-файл для выяснения того, как отправлять задания PostScript на принтер. Например, нет никакого смысла просить поддерживающий только одностороннюю печать черно-белый принтер, продаваемый в Америке, распечатать двухсторонний цветной документ на европейской бумаге формата B4.

Старые системы печати в UNIX и Linux не умели пользоваться PPD-файлами. Пользователям приходилось либо учиться преобразовывать свои PostScript-данные в подходящий формат, либо довольствоваться тем, что им предлагалось по умолчанию. Система CUPS наоборот изначально была создана так, чтобы пользоваться преимуществами этого богатого источника информации. На самом деле CUPS даже зависит от PPD-файлов.

Чтобы отыскать PPD-файлы, придется немного потрудиться. На локальной машине PPD-файл может находиться в каталоге `/etc/cups/ppd` или `/usr/share/cups/model`. Команда `locate .ppd` помогает отслеживать такие файлы. Что касается сетевых принтеров, их PPD-файлы, скорее всего, будут храниться удаленно; клиенты CUPS получают PPD-информацию от соответствующего CUPS-сервера.

PPD-файлы — это просто текстовые файлы. Не помешает заглянуть в один такой файл и посмотреть, какая информация в нем содержится.

У принтеров PostScript у всех есть поставляемые производителем PPD-файлы, которые можно найти либо на установочном диске, либо на Web-сайте производителя. PPD-файлы из распространяемой вместе с CUPS библиотеки хранятся в каталоге `/usr/share/cups/model`; а все используемые на текущий момент PPD-файлы CUPS копирует в каталог `/etc/cups/ppd`.

CUPS также прмиеняет PPD-файлы для описания принтеров, у которых нет интерпретатора PostScript. Весь трюк заключается в одном дополнительном поле. Взгляните:

```
$ grep cupsFilter /usr/share/cups/model/pxlmono.ppd
*cupsFilter: "application/vnd.cups-postscript 0 pstopxl"
```

Можно сравнить два тесно связанных PPD-файла (например, файл `pxlmono.ppd` и файл `pxlcolor.ppd`) при помощи команды `diff`, чтобы увидеть, чем именно отличаются два типа принтеров.

Если производитель принтера не предоставляет PPD-файл — возможно потому, что у принтера нет интерпретатора PostScript, и производителя не волнует ничего, кроме Windows — следует зайти на сайт `linuxprinting.org` и поискать информацию в базе данных Foomatic. Не исключено, что принтер поддерживается проектом Gutenprint

(gutenprint.sourceforge.net, ранее известным как Gimp-Print). Если доступны несколько или все перечисленные источники PPD-файлов, и пользователи хотят наилучшего возможного качества, нужно попробовать каждый вариант и посмотреть, в каком случае выходные данные выглядят лучше всего.

Если отыскать PPD-файл нигде не удается, тогда:

- вероятно, следовало проверить принтер по базе данных на сайте linuxprinting.org, прежде чем приобретать его;
- вполне вероятно, что существует какой-нибудь универсальный PPD-файл, который позволит выполнять печать, хотя и не будет пользоваться преимуществами всех функциональных возможностей принтера;
- в случае усовершенствования универсального PPD-файла до состояния, позволяющего ему работать лучше с данным принтером, следует обязательно отправить его в базу данных Foomatic.

Фильтры

Вместо того чтобы использовать специализированную утилиту печати для каждого принтера, CUPS использует ряд фильтров, которые преобразовывают распечатываемый файл в формат, понятный для имеющегося принтера.

Схема фильтров в CUPS является довольно элегантной. Когда CUPS получает подлежащий распечатке файл, она определяет его тип MIME, типы MIME, понимаемые имеющимся принтером, и фильтры, которые ей нужны для того, чтобы преобразовать первый в один из последних.

Для распознания типа поступающих данных CUPS пользуется правилами, указанными в файле /etc/cups/mime.types. Например, правило

```
application/pdf      pdf string (0,%PDF)
```

означает следующее: "Если файл имеет расширение pdf или начинается со строки %PDF, тогда его типом MIME является application/pdf".

О том, как преобразовать один тип данных в другой, CUPS узнает, просматривая правила в файле /etc/cups/mime.convs. Например, правило

```
application/pdf      application/postscript 33 pdftops
```

означает следующее: "Чтобы преобразовать файл application/pdf в файл application/postscript, нужно запустить фильтр pdftops". Число 33 — это стоимость такой операции преобразования.

В случае возникновения необходимости написать свои собственные фильтры (что маловероятно), не вздумайте изменять распространяемые файлы. Создайте дополнительный набор файлов со своим собственным базовым именем и поместите их в каталог /etc/cups, где CUPS сможет их найти. CUPS считывает все файлы с суффиксом .types и .convs, а не только файлы с суффиксом mime.types и mime.convs.

Последними компонентами в конвейере CUPS являются фильтры, которые взаимодействуют с принтером напрямую. В PPD-файле не поддерживающего PostScript принтера можно увидеть строки типа

```
*cupsFilter: "application/vnd.cups-postscript 0 foomatic-rip"
```

или даже

```
*cupsFilter: "application/vnd.cups-postscript foomatic-rip"
```

Строка, заключенная в кавычки, имеет точно такой же формат, как и строка в файле `mime.convs`, но в ней указан только один тип MIME, а не два. Она сообщает о том, что фильтр `foomatic-rip` преобразовывает данные типа `application/vnd.cups-postscript` в родной формат данных принтера. Стоимость равна нулю (или вообще опускается), потому что существует только один способ выполнить этот шаг, так что зачем притворяться, что есть еще какая-то стоимость? (PPD-файлы Gutenprint для не поддерживающих PostScript принтеров немного отличаются.)

Имея документ и целевой принтер, CUPS использует файлы `types`, чтобы определить тип документа. Далее она проверяет PPD-файл, чтобы выяснить, какой тип данных требуется принтеру. Потом она использует файлы `.convs`, чтобы составить все цепочки фильтров, которые могли бы преобразовать один тип данных в другой, и узнать, сколько каждая из них будет стоить. И, наконец, она выбирает самую дешевую по стоимости цепочку фильтров и передает документ через них.

Последний фильтр в цепочке передает уже преобразованные в пригодный для печати формат данные серверному компоненту, который пересылает их принтеру, используя любой понимаемый данным принтером протокол.

Узнать, какие фильтры доступны в системе, можно с помощью команды `locate pstops` (`pstops` — это популярный фильтр, который выполняет с PostScript-заданиями различные манипуляции, например добавляет PostScript-команды для указания количества копий. Остальные фильтры обязательно будут где-нибудь рядом).

Просмотреть список доступных серверных компонентов можно, выполнив команду `lpinfo -y`. Если в используемой системе нет серверного компонента для необходимого сетевого протокола, возможно, он доступен где-то в сети Internet или на сайте производителя.

23.4. Администрирование сервера CUPS

`cupsd` запускается во время загрузки и выполняется постоянно. Все рассматриваемые нами дистрибутивы Linux настроены работать именно так по умолчанию.

Конфигурационный файл CUPS называется `cupsd.conf`; обычно его можно найти в каталоге `/etc/cups`. По формату он похож на конфигурационный файл Apache. Если вы умеете работать с одним из этих файлов, значит, сможете работать и с другим.

После внесения изменений в этот конфигурационный файл следует выполнить команду `/etc/init.d/cups restart` для перезапуска демона и вступления внесенных изменений в силу. (В дистрибутивах Debian и Ubuntu вместо этой команды следует применять `/etc/init.d/cupsys restart`).

Конфигурационный файл CUPS можно редактировать вручную или, если установлена настольная среда KDE, с помощью приложения KDE Print Manager (Диспетчер печати KDE), которое доступно через окно центра управления KDE. Этот процесс детально описан в руководстве *KDEPrint Handbook*, где также можно найти всю необходимую информацию о переменных CUPS, их предназначении и их значениях по умолчанию.

Получить доступ к посвященному непосредственно системе CUPS разделу диспетчера печати KDE (KDE Print Manager) можно при помощи команды `cupsdconf`. Эта команда входит в состав пакетов `kdelibs` большинства систем, поэтому устанавливать всю среду KDE для того, чтобы иметь возможность пользоваться ею (т.е. этой командой), вовсе не обязательно.

Производственного опыта работы с графическим пользовательским интерфейсом KDE у нас нет, но когда мы его тестировали, он "жаловался" нам на то, что не понимает некоторые опции в создаваемых по умолчанию файлах `cups.conf`, на всех наших тестовых системах. В SUSE он вообще отказывался запускаться, очевидно из-за того, что строка

```
AuthType BasicDigest
```

в файле `cups.conf` вынуждала его искать несуществующий файл `/etc/passwd.md5`. (Другие системы использовали `AuthType Basic` как параметр по умолчанию.) А у вас вообще все может быть по-другому.

Настройка сервера печати

Чтобы заставить CUPS принимать задания на печать из сети, потребуется внести два изменения в файл `cupsd.conf`. Сначала нужно изменить

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
</Location>
```

на

```
<Location />
Order Deny,Allow
Deny From All
Allow From 127.0.0.1
Allow From сетевой_адрес
</Location>
```

На месте *сетевой_адрес* следует указать IP-адрес сети, из которой должны приниматься задания на печать (например, 192.168.0.0). Далее нужно отыскать ключевое слово `BrowseAddress` и указать напротив него адрес рассылки в этой сети и порт CUPS:

```
BrowseAddress 192.168.0.255:631
```

Эти два шага вынудят сервер принимать запросы с любой машины в сети и рассылать известную ему информацию об обслуживаемом им принтере всем имеющимся в сети демонам CUPS.

Вот и все! После перезапуска демон CUPS станет сервером.

Автоматическое конфигурирование принтеров

На самом деле система CUPS может использоваться и без принтера (например, для преобразования файлов в формат PDF или формат факса), но в основном она все-таки применяется для управления принтерами. В этом разделе речь пойдет о самих принтерах.

В некоторых случаях добавление принтера является обычным делом. CUPS старается автоматически обнаруживать USB-принтеры при их подключении и выяснять, что с ними нужно сделать.

Производители принтеров обычно предоставляют установочное программное обеспечение, которое автоматически (без участия пользователя) выполняет большую часть необходимых инсталляционных операций под управлением Windows и даже MacOS X (где тоже используется CUPS). Но рассчитывать на то, что производители позаботятся о предоставлении подобного установочного программного обеспечения для Linux, нельзя.

Даже если установку приходится выполнять самостоятельно, процесс добавления принтера часто состоит всего лишь из подключения оборудования, установки соединения с Web-интерфейсом CUPS (`localhost:631/admin`) и предоставлении ответов на несколько вопросов. KDE и GNOME поставляются со своим собственными утилитами для конфигурирования принтера, которые могут использоваться вместо интерфейса CUPS.

Если кто-то другой добавляет принтер, и об этом становится известно одному или более функционирующим в сети серверам CUPS, ваш сервер CUPS тоже уведомляется о появлении нового принтера. Ни добавлять этот принтер явно в локальный перечень устройств, ни копировать его PDD-файл на свою машину вам не нужно. Это все делается посредством зеркал.

Конфигурирование сетевых принтеров

Сетевым принтерам необходима какая-то своя собственная конфигурация только для того, чтобы быть членами сети TCP/IP. В частности, им необходимо знать свой IP-адрес и сетевую маску. Такая информация обычно передается им одним из двух способов.

Почти все современные принтеры могут получать эту информацию по сети от сервера BOOT или DHCP. Этот метод работает хорошо в средах с множеством однотипных принтеров. Более подробную информацию о DHCP можно найти в разделе 12.9.

Второй способ заключается в установке статического IP-адреса через консоль принтера, которая обычно состоит из ряда кнопок на передней панели и однострочной индикаторной панели. Все, что потребуется сделать — это походить по меню и отыскать в них раздел, где можно установить IP-адрес. (Если вдруг попадется опция, позволяющая распечатать меню, следует воспользоваться ею и сохранить печатную версию.)

Некоторые принтеры предоставляют доступ к виртуальной консоли через последовательный порт. Это замечательная идея, но в целом предполагает выполнение почти такого же количества операций, как и использование кнопок на передней панели принтера.

Если по-другому не получается, можно воспользоваться прилагаемым к принтеру руководством. (Очень многие принтеры поставляются вместе с неплохими руководствами.)

Будучи сконфигурированными, сетевые принтеры обычно имеют доступную через браузер “Web-консоль”. Однако для того чтобы с ними можно было работать через эту консоль, у них должен быть IP-адрес, т.е. возникает проблема самозагрузки.

Когда принтер уже находится в сети и готов к использованию, следует обязательно убедиться в том, что он защищен от несанкционированного доступа. О том, как это можно сделать, будет рассказываться чуть позже в этой главе, в разделе 23.7.

Примеры конфигурирования принтеров

В качестве примеров давайте добавим принтер с параллельным интерфейсом groucho и сетевой принтер fezmo из командной строки:

```
# lpadmin -p groucho -E -v parallel:/dev/lp0 -m pxlcolor.ppd
# lpadmin -p fezmo -E -v socket://192.168.0.12 -m laserjet.ppd
```

Как видите, groucho предоставляется порт `/dev/lp0`, а femzo — IP-адрес 192.168.0.12. Мы указываем каждое устройство в виде универсального идентификатора ресурса (URI) и выбираем PDD-файл из PDD-файлов в каталоге `/usr/share/cups/model`.

Если локальный демон cupsd сконфигурирован как сетевой сервер, он сразу же сделает эти новые принтеры доступными для других клиентов в сети.

Вместо интерфейса командной строки можно использовать и предлагаемые CUPS Web-средства конфигурирования. Это касается всех описываемых в этом разделе задач по администрированию.

CUPS принимает множество разнообразных URI-идентификаторов принтеров. Вот еще несколько примеров:

```
ipp://zoe.canary.com/ipp
lpd://riley.canary.com/ps
serial://dev/ttyS0?baud=9600+parity=even+bits=7
socket://gillian.canary.com:9100
usb://XEROX/Phaser%206120?serial=YGG210547
```

Некоторые URI-идентификаторы принимают опции (вроде serial), а некоторые — нет. Команда lpinfo -v отображает список устройств, которые система может видеть, и список типов URI-идентификаторов, которые понимает система CUPS.

Создание класса принтеров

"Класс" — это группа принтеров, которые используют очередь совместно. Задания в очереди могут распечатываться на любом из этих принтеров, а именно на том, который становится доступен первым. Показанная ниже команда создает класс haemer и добавляет в него три принтера: riley, gilly и zoe.

```
# lpadmin -p riley -c haemer
# lpadmin -p gilly -c haemer
# lpadmin -p zoe -c haemer
```

Обратите внимание на то, что никакого явного шага для создания класса нет; класс существует только при условии добавления в него принтеров. На самом деле интеллектуальные возможности CUPS этим не ограничиваются: если множеству принтеров в сети присваивается одинаковое имя, CUPS воспринимает их как неявный класс и автоматически распределяет между ними связанную с заданиями нагрузку.

Отключение принтеров для прохождения технического осмотра

В случае возникновения потребности удалить принтер или класс, это можно легко сделать при помощи команды lpadmin -x:

```
# lpadmin -x fezmo
# lpadmin -x haemer
```

Но как быть, если нужно только временно отключить принтер для прохождения технического осмотра, а не удалять его? В таком случае можно просто заблокировать очередь печати на одном из концов. Если отключить "хвост" (т.е. выходную или принтерную сторону) очереди, пользователи по-прежнему смогут отправлять задания, но эти задания никогда не будут печататься. Если отключить "головную" (входную) часть очереди, те задания, которые уже находятся в очереди, будут распечатаны, а вот все попытки добавить в очередь новые задания будут отклоняться.

Команды disable и enable контролируют выходную сторону очереди, а команды reject и accept — ее принимающую сторону. Например:

```
# disable groucho
# reject corbet
```

Какую же из них лучше использовать? Принимать задания на печать, которые точно не смогут быть распечатаны в ближайшем будущем, глупо, поэтому для длительных периодов простоя лучше использовать команду `reject`. Для коротких перерывов, которые даже не должны быть заметны пользователям (например, когда нужно просто удалить застрявшую в принтере бумагу), лучше использовать команду `disable`.

Администраторы иногда просят дать им какую-нибудь подсказку, которая помогла бы им запомнить, какие команды контролируют каждый конец очереди. Предлагаем попробовать такой вариант: если CUPS "отклоняет" (`reject`) задание, это означает, что его нельзя "предоставить". Еще один способ не путать команды — это запомнить, что приниматься (`accept`) и отклоняться (`reject`) могут только задания на печать, а отключаться (`disable`) и включаться (`enable`) — только принтеры.

Предостережение: команда `enable` является не только командой системы CUPS, но и также встроенной интерпретатора команд `bash`. Если не указывается полный путь к команде (`/usr/bin/enable`), интерпретатор `bash` считает, что имеется в виду ее собственная команда `enable`. Команда `enable` интерпретатора `bash` включает и отключает команды, встроенные в `bash`, так что ее можно использовать и для отключения самого интерпретатора[2]:

```
$ enable -n enable
```

Система CUPS иногда сама временно отключает принтер, с которым не может нормально работать (например, из-за того, что кто-то выдернул его кабель). Устранив проблему, следует снова активизировать очередь. Если забыть это сделать, `lpstat` напомнит. (Более подробную информацию по этой теме и альтернативному подходу можно найти по адресу `www.linuxprinting.org/beh.html`.)

Другие задачи, связанные с конфигурированием

Сегодняшние принтеры имеют очень много опций, которые можно конфигурировать. CUPS позволяет настраивать самые разные функциональные возможности с помощью ее Web-интерфейса и посредством команд `lpadmin` и `lpoptions`. Как правило, команда `lpadmin` используется для выполнения задач на уровне системы, а команда `lpoptions` — для выполнения задач на уровне пользователя.

Команда `lpadmin` позволяет ограничивать доступ более точным образом, чем команда `disable` и команда `reject`. Например, с ее помощью можно создавать квоты на печать и указывать, на каких именно принтерах разрешается выполнять печать тем или иным пользователям.

Форматы бумаги

В США и Канаде наиболее распространенный формат бумаги называется "Letter" (письмо) и равняется 8,5×11 дюймам. Некоторые дистрибутивы Linux (такие как Knopix и SUSE) выпускаются в Европе, где никто даже не знает, что такое дюймы, или в Англии, где знают, что такое дюймы, но не пользуются ими для измерения бумаги.

[2] Чтобы заработать дополнительные очки, попробуйте догадаться, как можно снова включить встроенную в `bash` команду `enable` теперь, когда доступ к ней заблокирован. Команда `enable enable` ничего не даст!

В этих странах и в Японии самый распространенный формат бумаги называется "A4", и все принтеры поставляются с лотками размера A4. Поэтому утилиты печати в некоторых дистрибутивах генерируют изображения размером со страницу формата A4 по умолчанию.

Лист формата A4 является удобным, потому что он не пропорционален — имеется в виду, с математической точки зрения. Соотношение длины к ширине у листа формата A4 равняется $\sqrt{2}$. Если сложить лист A4 пополам горизонтально, получится два листа с одинаковым соотношением длины к ширине. Такой формат листа уже называется "A5". Если сложить пополам лист формата A5, получится два листа формата A6. Если пойти в обратном направлении: лист формата A3 будет по площади в два раза больше листа формата A4, но точно такой же формы, и т.д.

Другими словами, можно изготавливать листы формата A0, по площади равняющиеся одному квадратному метру, и с помощью бумагорезальной машины создавать из них листы других необходимых размеров. Единственным распространенным в США форматом бумаги, с которым можно проворачивать подобное, является формат "Ledger" ("гроссбух", 11×17 дюймов, также известный как "таблоид"): если сложить лист такого формата пополам, получится два листа формата "письмо".

Также еще существуют ISO-форматы серии B и серии C, которые сохраняют пропорцию $1:\sqrt{2}$, но имеют другую базовую площадь. B0 составляет 1 метр в высоту, а C0 имеет площадь $2^{1/4}$ м^2. Инженеры сразу же увидят, что стороны листа B*n* представляют собой среднее геометрическое сторон A$n-1$ и A*n*, а стороны листа C*n* — среднее геометрическое сторон A*n* и B*n*.

Что это все означает? B*n* выглядит точно так же, как A*n*, но является больше, а C*n* представляет собой нечто среднее между ними двумя. Отчет на листе формата A4 прекрасно поместится в манильскую папку формата C4. Если сложить по середине письмо формата A4, оно превратиться в письмо формата A5, которое легко войдет в конверт формата C5. А если его сложить еще раз, оно также легко войдет и в конверт формата C6.

Немного усложняет дело то, что в Японии имеется своя серия форматов B, которые чуточку отличаются. Хоть они и имеют те же пропорции, что и ISO-листы, японский формат B4 представляет собой среднее арифметическое форматов A3 и A4, что делает его немного больше ISO-формата B4. Японских форматов серии C не существует.

Система ISO позволяет запросто копировать две страницы формата B5 на одну страницу формата B4, а также печатать по сколько угодно уменьшенных изображений страниц на одном листе. На европейских копировальных машинах часто имеются кнопки, позволяющие уменьшать или увеличивать изображение на множитель $\sqrt{2}$.

Если в системе установлена команда `paperconf`, можно использовать ее и сделать так, чтобы размеры разных известных форматов бумаги отображались в дюймах, сантиметрах или точках (каждая из которых составляет $^1/_{72}$ дюйма). Для американцев в табл. 23.1 перечислены некоторые наиболее распространенные случаи использования основных форматов бумаги.

К сожалению, бумага формата A4 немного тоньше и длиннее (8,3×11,7 дюймов) бумаги американского формата "Letter". Печать документа формата A4 на бумаге формата "Letter" обычно заканчивается обрезанием таких его важных частей, как верхние колонтитулы, нижние колонтитулы и номера страниц. И наоборот, печать документа американского формата на бумаге формата A4 приводит к усечению его сторон (хотя это и менее страшно).

Таблица 23.1. Основные случаи использования ISO-форматов бумаги

Форматы	В основном используются для:
A0, A1	постеров (афиш)
A3, B4	газет
A4	универсальных документов
A5	блокнотов (примерно 5x8 дюймов)
B5, B6	книг, открыток, немецкой туалетной бумаги
A7	учетных карточек “3x5”
B7	паспортов (даже в США паспорта имеют формат B7)
A8	визитных карточек
B8	игральных карт

У некоторых программных пакетов могут быть свои собственные параметры по умолчанию для формата бумаги. Например, GNU-утилита `enscript` обслуживается в Финляндии человеком по имени Маркку Росси (Markku Rossi) и по умолчанию использует формат A4. Те, кто проживает в Америке, и у кого дистрибутив не скомпилировал `enscript` с другом параметром по умолчанию, могут сделать только одно — взять исходный код и переконфигурировать его. Однако обычно проще поступить следующим образом: указать нужный формат (размер) бумаги в командной строке или конфигурационном файле GUI. Если документы печатаются с обрезанными концами или сторонами, причиной, скорее всего, является несовместимость форматов бумаги.

Также бывает возможным настроить используемый по умолчанию формат бумаги для многих связанных с печатью задач при помощи команды `paperconfig`, переменной среды PAPERSIZE или содержимого файла `/etc/papersize`. (Обратите внимание: `paperconfig` — не то же самое, что `paperconf`.)

Команды совместимости

В прежние времена существовало две системы печати, которые конкурировали между собой: одна использовалась в системах BSD UNIX, а другая — в UNIX-системах System V. Каждая из них поддерживала относительно простые очереди на печать и предоставляла команды для создания, удаления, запуска, остановки и приостановки очередей, а также для добавления или удаления из очереди отдельных заданий.

У некоторых может возникнуть желание спросить, почему систем было две, и существовали ли между ними хоть какие-нибудь значительные отличия. Встаньте посредине собрания пользователей Linux и выкрикните: “Любой, кто пользуется `vi` — идиот!”. После этого можете прийти и спросить нас снова.

На всякий случай CUPS поставляется с командами совместимости, заменяющими обе предыдущих системы. Отчасти это было сделано для того, чтобы заманить любителей “старины”, привыкших пользоваться прежними системами, но главной задачей, конечно же, было обеспечить совместимость с существующим программным обеспечением.

Чтобы не возникало никаких недоразумений, необходимо отметить, что эти команды не всегда делают все то же, что делали их исходные версии, а некоторые используемые реже и зависящие от производителя команды вообще еще не были реализованы. Тем не

менее, многие сценарии, в которых присутствуют эти команды, работают с CUPS ничуть не хуже. Отсутствие тех или иных возможностей стоит воспринимать как шанс: хочется внести вклад в дело мира или развитие теории оптимальности Парето, можно написать свой код.

В табл. 23.2 перечислены все CLI-команды, которые поставляются с CUPS, и системы, в которых они использовались первоначально. Для удобства они были поделены на три категории по источнику, из которого они произошли.

Таблица 23.3. Поставляемые с CUPS утилиты командной строки и системы, в которых они использовались первоначально

	Команда	Функция
CUPS	`lpinfo`	Показывает доступные устройства или драйверы
	`lpoptions`	Отображает или устанавливает опции и параметры по умолчанию принтера
	`lppasswd`	Добавляет, изменяет или удаляет пароли дайджеста
	`cupsdconf`[a]	Представляет собой утилиту для конфигурирования
	`cups-config`[a]	Выводит информацию об API-интерфейсе, компиляторе, каталоге и канале связи системы CUPS
SystemV	`lp`	Печатает файлы
	`cancel`	Отменяет задания
	`accept, reject`	Принимает или отклоняет поступающие в очередь задания
	`disable, enable`	Останавливает или запускает принтеры и классы
	`lpstat`	Печатает информацию о состоянии CUPS
	`lpadmin`	Конфигурирует принтеры и классы CUPS
	`lpmove`	Перемещает задание в новое место
BSD	`lpr`	Печатает файлы
	`lprm`	Отменяет задания на печать
	`lpq`	Отображает статус очереди принтера
	`lpc`	Представляет собой общую программу для управления принтерами

[a] Не путайте эти похожие имена. `cupsdconf` — это утилита с графическим интерфейсом пользователя (GUI), поставляемая с KDEPrint, а `cups-config` — это утилита с интерфейсом командной строки (CLI), поставляемая с CUPS.

Основные программы печати

Печать — это не только спулинг и печатные задания. Даже в системе Ubuntu команда

```
$ man -k . | egrep -i 'ghostscript|cups|print(er|ing| *(job|queue|filter))'
```

отображает более 88 касающихся печати страниц из руководства — и это при поиске "на скорую руку". (Кстати, изучая связанные с печатью команды, обратите внимание на то, что команда `print` не имеет совершенно никакого отношения к печати.) О некоторых из этих команд и утилит стоит знать.

`pr` — это одна из наиболее старых утилит печати. Она изменяет формат текстовых файлов так, чтобы он соответствовал формату печатной страницы. В частности, она раз-

бивает содержащиеся в них данные на включающие по 66 строк страницы, добавляет верхние и нижние колонтитулы и, при необходимости, устанавливает между строками двойной интервал. Она прекрасно подходит для выполнения небольших преобразовательных операций над текстовыми файлами на их пути к принтеру.

Adobe-команда `enscript` выполняет похожие преобразования с несколькими дополнительными усовершенствованиями; она тоже выводит данные в формате PostScript. GNU-команда `enscript` представляет собой распространяемую с открытым исходным кодом версию этой команды и имеет обратную совместимость с аналогичной командой Adobe; однако GNU-команда `enscript` предлагает целый ряд новых функциональных возможностей, среди которых чувствительное к языку выделение, поддержка для различных форматов (размеров бумаги), загрузка шрифтов и определяемые пользователем заголовки.

Одним из главных преимуществ `enscript` была реализация возможности печати сразу двух страниц на одном листе. Те, кто до сих пор пользуется `enscript` из-за этой возможности, могут попробовать использовать в CUPS опцию `-o number-up=2` с командой `lpr`.

Одной из самых сложных утилит считается утилита Ghostscript, которую первоначально создал Л. Питер Дойч (L. Peter Deutsch) для того, чтобы иметь возможность печать документы PostScript на недорогих принтерах PCL. Сегодня Ghostscript интерпретирует не только PostScript, но и PDF. CUPS использует ее в качестве фильтра, но Ghostscript также может создавать изображения страниц для экрана как самостоятельно, так и с помощью программ, подобных `gv`, GNOME Ghostview (`ggv`) или KDE KGhostView.

Абсолютно все дистрибутивы Linux поставляются с бесплатной версией Ghostscript, подробнее о которой можно узнать на сайте `www.ghostscipt.com`. Коммерческая версия Ghostscript с поддержкой доступна на сайте компании Artifex Software.

Документация по системе CUPS

Документации по системе CUPS в принципе хватает, но иногда, чтобы добыть ее приходится прикладывать определенные усилия. Описания, приводимые на страницах руководства, как например описания для команды `lpr`, могут быть поверхностными. Отыскав что-нибудь на странице руководства, не думайте, что вы такое сделать не сможете; сначала попробуйте поискать то же самое в Google.

Установка CUPS поставляется с несколькими руководствами в PDF- и HTML-формате. Один из способов увидеть их — это подключиться к серверу CUPS и щелкнуть на ссылке, указывающей на раздел доступной оперативной справки. К сожалению, если установить соединение с сервером CUPS не возможно, ничего увидеть не получится.

Ту же самую документацию можно найти на сайте `www.cups.org`. Она должна быть также доступна в каталоге `/usr/share/doc/cups`. Если ее там нет, можно попробовать выполнить следующую команду:

```
$ locate doc | grep cups
```

Еще один вариант — запросить диспетчер пакетов вашего дистрибутива.

23.5. Советы по выявлению проблем

Никогда не следует забывать перезапускать демон `cupsd` после внесения изменений в его конфигурационный файл. Самый удобный способ перезапустить его — выполнить

команду `/etc/init.d/cups restart` (или команду `/etc/init.d/cupsys restart`, если речь идет о системе Debian или Ubuntu). Также его можно перезапустить и через приложение KDE Print Manager (Диспетчер печати KDE). В теории еще можно отправить `cupsd` сигнал HUP, но похоже, что на системах SUSE это приводит только к уничтожению демона.

Ведение журналов в CUPS

CUPS поддерживает три журнала: журнал страниц, журнал доступа и журнал ошибок. Журнал страниц представляет собой просто список печатавшихся страниц. Журнал доступа и журнал ошибок похожи на аналогичные журналы в Apache. И в этом нет ничего удивительного, потому что CUPS-сервер — это Web-сервер.

Уровень регистрации и путь к журнальным файлам задается в файле `cupsd.conf`. Обычно журнальные файлы сохраняются в каталоге `/var/log`.

Ниже приведен фрагмент из журнального файла, охватывающий одно задание на печать:

```
I [26/Jul/2006:18:59:08 -0600] Adding start banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Adding end banner page "none" to job 24.
I [26/Jul/2006:18:59:08 -0600] Job 24 queued on 'Phaser_6120' by 'jsh'.
I [26/Jul/2006:18:59:08 -0600] Started filter /usr/libexec/cups/filter/pstops
     (PID 19985) for job 24.
I [26/Jul/2006:18:59:08 -0600] Started backend /usr/libexec/cups/backend/usb
     (PID 19986) for job 24.
```

Проблемы с прямой печатью

Удостовериться в наличии физического соединения с локальным принтером можно напрямую, запустив программу обслуживания принтера. Например, вот что мы получим, если запустим программу обслуживания принтера, который подключается через USB-кабель:

```
$ /usr/lib/cups/backend/usb
direct usb "Unknown" "USB Printer (usb)"
direct usb://XEROX/Phaser%206120?serial=YGG210547 "XEROX Phaser 6120"
   "Phaser 6120"
```

Если USB-кабель вдруг случайно кто-то выдернет (или повредит), строка для этого принтера исчезнет из вывода программы обслуживания:

```
$ /usr/lib/cups/backend/usb
direct usb "Unknown" "USB Printer (usb)"
```

Проблемы с сетевой печатью

Прежде чем начинать искать проблему печати в сети, проверьте, получается ли выполнять печать с компьютера, к которому непосредственно подключен принтер. Ваша “проблема с сетевой печатью” может оказаться просто “проблемой печати”. Также проверьте, все ли в порядке с сетью.

Далее попробуйте установить соединение с обслуживающим данный принтер демоном `cupsd` при помощи браузера (*имя_хоста*`:631`) или команды `telnet` (`telnet` *имя_хоста* `631`).

В случае возникновения проблем при настройке соединения с сетевым принтером, помните о том, что на клиентском компьютере у вас должна быть очередь для задания, способ для решения того, куда отправлять задание, и метод отправки задания на компьютер, который отвечает за обслуживание очереди на печать. На сервере печати у вас должно быть место для постановки задания в очередь, соответствующие права доступа для разрешения печати задания, а также способ для вывода данных на устройство.

Чтобы выяснить причину этих проблем, возможно, придется заглянуть в несколько мест, а именно в:

- системные журнальные файлы на клиентском компьютере (в случае проблем с распознаванием имен и правами доступа);
- системные журнальные файлы на сервере печати (в случае проблем с правами доступа);
- журнальные файлы CUPS на клиентском компьютере (если отсутствуют какие-то фильтры или каталоги, либо если имеются неизвестные принтеры, и т.д.);
- журнальные файлы на сервере печати (в случае появления сообщений о неправильных именах устройств, неправильных форматах и т.д.).

Путь к журнальным файлам системы указан в файле `/etc/syslog.conf`. Путь к журнальным файлам CUPS указан в файле `/etc/cups/cupsd.conf`.

Проблемы с дистрибутивами

CUPS все еще совершенствуется, и исправления ошибок выходят довольно часто. Некоторые проблемы бывают хуже других, а некоторые имеют отражающиеся на безопасности последствия. Например, в более старых версиях Red Hat, CUPS постоянно взламывалась. Наиболее правильным решением для таких систем является обновление ОС. Но если нет возможности установить более новую версию Red Hat или Fedora, можно попробовать установить текущую версию CUPS.

Компания Easy Software Products (ESP) продает коммерческую версию CUPS, которая называется “ESP PrintPro” и поддерживает гораздо больше моделей принтеров, чем бесплатная версия. Если требуется поддержка для какого-нибудь необычного принтера, и отыскать необходимые драйверы в Web не удается, возможно, они есть у ESP. Кроме того, ESP также оказывает платные услуги. Дополнительную информацию можно найти по адресу `www.easysw.com`.

23.6. Практические советы касательно принтеров

При работе с принтерами неизбежно возникают различные проблемы и сбои. Ниже приводится несколько советов по поводу того, как можно свести их количество к минимуму. Если ни один из них не поможет, радуйтесь хотя бы тому, что не доводится пользоваться матричным принтером, подключающимся через последовательный порт RS-232. Если, конечно, это так.

Выбор принтера

Прежде чем покупать принтер или брать “бесплатный” принтер, который кто-то другой выкидывает, следует заглянуть в базу данных Foomatic на сайте `linuxprinting.org` и проверить, насколько хорошо он поддерживается под Linux. В этой базе данных все

принтеры поделены на четыре категории, начиная от категории Paperweight (Пресс-папье) и заканчивая категорией Perfectly (Отлично); выбирать нужно категорию Perfectly (Отлично).

CUPS "любит" PostScript-принтеры. Конфигурирование этих принтеров обычно выполняется очень легко.

Не-PostScript принтеры тоже поддерживаются, но не так хорошо. Чтобы выполнять печать на этих принтерах, потребуется специальное программное обеспечение, умеющее преобразовывать задания на печать в предпочитаемый принтером формат PDL или другой формат данных. Не исключено, что такое программное обеспечение удастся найти в пакете CUPS или в одном из других упоминавшихся ранее в этой главе источников.

GDI-принтеры

Windows по-прежнему обладает двумя преимуществами, одним из которых является ее поддержка для очень старых (маломощных) принтеров. Дешевые принтеры, используемые на системах Windows, вместе называются *GDI-принтерами* или *принтерами WinPrinters*. Такие принтеры обладают очень незначительным количеством встроенных логических возможностей и не имеют никаких интерпретаторов для PDL. Они ожидают, что растеризация будет выполняться на обслуживающем их компьютере.

Часть информации, необходимой для взаимодействия с GDI-принтерами, скрыта в специальном, патентованном коде Windows. Такая секретность затрудняет разработку Linux-поддержки для таких устройств, но сообщество сторонников ПО с открытым кодом продемонстрировало удивительную способность к реконструированию. CUPS поддерживает многие из принтеров WinPrinters.

Вторым преимуществом Windows является ее поддержка для новейших принтеров. Точно так же, как новые видео- и аудиокарты, новые принтеры сначала выпускаются с драйверами для Windows, которые полностью поддерживают все документированные и недокументированные функциональные возможности данной модели. Драйверы для Linux обычно выходят с некоторым опозданием. В случае покупки нового модного принтера с расширенными возможностями, не исключено, что некоторое время им можно будет пользоваться только из Windows.

Двусторонняя печать

Дуплексор — это компонент аппаратного обеспечения, которые позволяет принтеру выполнять печать на обеих сторонах страницы. У некоторых принтеров он входит в комплектацию, а у некоторых идет как дополнение.

В случае, когда нет принтера, умеющего выполнять двустороннюю печать, и средств на его приобретение тоже, можно поступить следующим образом: распечатать на листах бумаги сначала только нечетные страницы, а затем перевернуть листы и распечатать только четные страницы. Однако сначала следует поэкспериментировать с двусторонним документом, чтобы узнать, как нужно правильно переворачивать листы бумаги, и только потом давать соответствующие инструкции принтеру.

Многие из программ печати могут помочь с этим; например, в программе Ghostview (`gv`) есть пиктограммы, позволяющие отмечать подлежащие печати страницы (четные или нечетные), и опция, позволяющая распечатывать только отмеченные страницы. CUPS-версии `lp` и `lpr` позволяют делать тоже самое при помощи опций `-o page-set=odd` и `-o page-set=even`. В случае частого использования эти опции можно сохранить в "экземпляре принтера", о котором более подробно рассказывалось в разделе 23.3.

Некоторые принтеры, особенно недорогие лазерные принтеры, разрабатывались без возможности двусторонней печати. Их производители часто предупреждают о непоправимом ущербе, которым может закончиться для принтера печать на обеих сторонах страницы. Мы никогда не слыхали о том, чтобы такой ущерб действительно имел место, но вряд ли производители стали бы предупреждать нас о нем просто так. А может стали бы?

Другие аксессуары для принтера

Помимо дуплексоров, многие принтеры позволяют добавлять память, дополнительные лотки для бумаги, жесткие диски другие аксессуары. Эти дополнительные компоненты могут давать возможность распечатывать задания, печать которых в противном случае оказалась бы невозможной, или просто делать печать заданий более эффективной. При наличии проблем с отправкой заданий на печать, следует просмотреть журналы ошибок и выяснить, не поможет ли справиться с этой проблемой увеличение памяти. О журналах ошибок более подробно рассказывалось в разделе 23.5.

Принтеры с последовательным и параллельным интерфейсом

Если принтер подключается прямо к компьютеру через кабель, значит, он использует для соединения либо последовательный, либо параллельный порт.

Хотя стандарт параллельных интерфейсов немного устарел, он все-таки подразумевает использование портов, конфигурирование которых требует приложения минимального количества усилий. Те, кто приобрел принтер с параллельным интерфейсом, скорее всего, смогут установить его без проблем — конечно, если у них на компьютере предусмотрен параллельный порт.

Последовательным портом на оборудовании Mac может служить порт FireWire, но в мире Linux в роли такого порта обычно выступает порт USB. Узнать больше об этом можно заглянув в базу данных о поддерживаемых USB-устройствах на сайте `www.qbik.ch/usb/devices` или `www.linux-usb.org`.

Вероятность приобретения давно устаревшего принтера, подключаемого через последовательный порт RS-232, очень низка. Но если такое все-таки случилось, при его конфигурировании придется приложить массу дополнительных усилий. Для того чтобы спулер мог нормально работать с принтером, ему необходимо знать скорость двоичной передачи в бодах (baud rate) и другие требующиеся для взаимодействия через последовательный порт параметры. Указать все эти параметры можно только в URI-идентификаторе устройства. (Этот процесс подробно описан в онлайновом руководстве *CUPS Software Administrators Manual* (Руководство для администраторов системы CUPS)). Пожалуй, легче будет купить другой принтер, чем вычислить точно все параметры, необходимые для того, чтобы заставить работать данный принтер с последовательным интерфейсом.

Сетевые принтеры

Многие принтеры имеют полнофункциональные сетевые интерфейсы, которые позволяют им находиться прямо в сети и принимать задания через одну и более сетей или протоколов печати. На принтеры, подключенные к сети, данные могут отправляться гораздо быстрее, чем на принтеры, подключенные к последовательному или параллельному порту.

23.7. Другие советы касательно принтеров

Некоторые связанные с печатью административные вопросы "выходят за пределы" особенностей Linux и CUPS. По большей части эти вопросы возникают потому, что принтеры являются механическими устройствами с характером, которые обходятся в определенную сумму денег каждый раз, когда их используют.

Используйте титульные страницы только в случае необходимости

CUPS может распечатывать для каждого задания начальные и заключительные страницы с названием этого задания и именем отправившего его на печать пользователя. Такие титульные страницы в некоторых случаях бывают полезными, потому что позволяют разделять задания, распечатываемые на одном и том же принтере разными пользователями, но в большинстве случаев они все-таки являются пустой тратой времени, красящего порошка и бумаги.

Мы рекомендуем отключать титульные страницы на глобальном уровне либо при помощи соответствующей опции в графическом пользовательском интерфейсе CUPS (либо путем выполнения команды `lpadmin` в ее интерфейсе командной строки) и затем включать их для отдельных заданий, которые действительно смогут от этого выиграть:

```
$ lpr -o job-sheets=confidential gilly.ps
```

Еще можно включать титульные страницы для отдельных пользователей путем использования команды `lpoptions`. Другой вариант — создать экземпляр принтера, добавляющий в задания титульные страницы. (О том, что такое экземпляр принтера, рассказывалось в разделе 23.3.)

При необходимости также можно создавать специальные титульные страницы, копируя одну из существующих титульных страниц из каталога `/usr/share/cups/banners`, изменяя ее нужным образом и сохраняя в том же каталоге с каким-нибудь новым именем.

Обеспечивайте утилизацию отходов

Все виды компьютерной бумаги пригодны к переработке. Коробки, в которых поставляется бумага, запросто можно использовать в качестве контейнеров для ненужных листов; в них бумага будет помещаться просто идеально. Все, что нужно сделать — это пометить эти коробки соответствующим образом, чтобы в них не бросали посторонние предметы (вроде скрепок, скобок скоросшивателей или газет).

Устанавливайте программы для предварительного просмотра

Пользователи часто распечатывают документ, обнаруживают какую-нибудь незначительную ошибку и затем начинают перепечатывать его. Такого абсолютно ненужного расхода бумаги и времени можно избежать, установив специальное программное обеспечение, позволяющее пользователям просматривать на экране монитора, как будет выглядеть их документ в распечатанном виде.

Просто установить программы для предварительного просмотра не достаточно; пользователи должны знать, как ими пользоваться. Обычно они совсем не против научиться. Ведение учета записей позволяет выявлять случаи, в которых один и тот же документ распечатывался повторно, что иногда может служить прекрасным указателем на пользователя, который не умеет пользоваться программами предварительного просмотра. Средства предварительного просмотра встроены во многие современные WYSIWYG-редакторы, браузеры и агрегаторы заданий на печать. Что касается других типов документов, возможны разные варианты. Произвольные документы формата PostScript и PDF можно просматривать с помощью такой программы, как Ghostview (`gv`). Что касается `roff`-документов, можно использовать утилиту `groff` и отправлять ее вывод программе Ghostview; для документов формата TeX можно попробовать использовать `xdvi`, `kdvi` или Evince.

Покупайте дешевые принтеры

Современный уровень технологий принтерного оборудования довольно высок, поэтому тратить много денег, чтобы получить хорошее качество и надежную механику, не нужно.

Не гоняйтесь за дорогими принтерами "для рабочих групп", если в этом нет острой необходимости. Если печатать нужно только текст, недорогой "персональный" принтер вполне может обеспечивать приемлемое качество и при этом быть почти таким же быстрым и надежным, да и к тому же более легким на целых несколько килограммов. Принтер со скоростью печати 10 страниц в минуту способен обслуживать около пяти штатных клерков или редакторов, так что для группы из 25 сотрудников, пожалуй, будет выгоднее купить пять принтеров по $250, чем один — за $1250.

В целом, не стоит покупать принтер (так же, как и жесткий диск или карту памяти) у производителей компьютеров. Их принтеры обычно представляют собой обычные принтеры, просто выступающие под их маркой и потому стоящие в два раза больше. Лучше покупать PostScript-принтеры, изготовленные для рынка ПК и Macintosh и продаваемые отдельно. (Некоторые компании, такие как HP, производят и компьютеры, и принтеры. С такими компаниями можно иметь дело.)

Даже если вы предпочитаете пользоваться продуктами известных марок, знайте, что ни в одной из них нельзя быть уверенным полностью. Например, лазерные принтеры производства HP прекрасно показали себя во время нашего тестирования. Они считаются заслуживающими доверия продуктами, да и компания HP известна как активно поддерживающая Linux и CUPS. Но несмотря на это, некоторые принтеры производства компании HP оказались для нас настоящим кошмаром. Поэтому прежде чем покупать принтер, обязательно ознакомьтесь с доступными на него в Internet отзывами.

Здесь тоже дешевизна является преимуществом: оправиться от "ошибки" в $250 гораздо легче, чем от той, что обошлась в $1250.

Всегда держите под рукой дополнительные картриджи

Лазерные принтеры периодически требуют замены картриджей. Появление блеклых или пропущенных фрагментов на странице свидетельствует о том, что у принтера заканчивается красящий порошок. Сменные картриджи лучше приобретать заранее. В крайнем случае можно попробовать вытащить картридж из принтера и слегка потрясти его, чтобы распределить равномерно остатки красящего порошка в нем. Нередко после этого на принтере удается распечатать еще сотню страниц.

Полоски и точки, скорее всего, означают, что нужно прочистить принтер. У принтера должна быть позволяющая сделать это кнопка или опция. Если ее нет или если она не помогла, следует внимательно прочитать приводимые производителем инструкции по выполнению прочистки или пригласить специалиста, чтобы он сделал это, и оплатить его услуги.

Производителям принтеров очень не нравится, когда используют старые или нелицензионные картриджи, и поэтому они делают все, что бы помешать этому. Во многих устройствах используются расходные материалы со специальными идентификационными номерами, которые распознаются принтером электронным или физическим образом. Даже если два принтера выглядят похоже (как принтер Xerox Phaser 6120 и принтер Konica-Minolta Magicolor 2450), это вовсе не обязательно означает, что на них обоих можно пользоваться одинаковыми картриджами.

Иногда "хирургическое вмешательство" позволяет делать картриджи одного производителя пригодными для принтера другого производителя, но только при условии, что этим занимается человек, который разбирается в том, что он делает. При неумелом обращении такой подход часто приводит к каким-нибудь не очень приятным результатам, например к рассыпанию красящего порошка. В случае рассыпания красящего порошка следует воспользоваться пылесосом, после чего удалить остатки при помощи тряпки и холодной воды. Несмотря на общее мнение, красящий порошок, используемый в лазерных принтерах, не представляет угрозы ни для здоровья человека, ни для окружающей среды, но, как и другие качественные порошки, его все-таки лучше не вдыхать.

При замене картриджа, сохраняйте коробку и пакет, в которые завернут новый картридж, чтобы использовать их для заворачивания картриджа, который закончился. В телефонной книге или Internet всегда можно найти компанию, которая с радостью заберет у вас старый картридж.

Картриджи с номерами привели к появлению компаний, которые заново заполняют старые картриджи красящим порошком и потом продают их по более дешевой по сравнению с новыми картриджами цене. Эти компании также обычно не только "заправляют" старые картриджи, но еще обменивают их на уже заправленные картриджи, так что вы можете не только отдать свои старые картриджи на заправку, но и сразу же получить уже заправленные картриджи.

Мнения по поводу качества и срока службы заправленных картриджей расходятся. Например, одна известная нам компания, занимающая заменой картриджей, никогда не соглашается заправлять цветные картриджи или продавать переделанные картриджи, потому что считает, что количество расходов, требуемых на обслуживание использующих такие картриджи принтеров, значительно превышает количество сэкономленных подобным образом средств.

Обращайте внимание на то, во сколько обходится печать каждой страницы

Производители принтеров пользуются тем, что магистры делового администрирования (MBA) называют "измерением количественных показателей" для приведения общей стоимости продукта (насколько это возможно) в соответствие с объемом пользы, которую клиент может получить от него. Именно поэтому красящий порошок и чернила чаще всего стоят очень дорого, а модное принтерное оборудование порой продается по цене ниже стоимости его производства.

Например, на момент написания этой книги цветной лазерный принтер можно приобрести за $299. Полный набор сменных картриджей к нему стоит $278. Струйный принтер можно купить и за меньше чем $50, но сменные чернильные картриджи для него понадобятся гораздо раньше, а стоят они больше самого принтера.

У некоторых это может вызвать чувство возмущения, но должны же компании, занимающиеся производством принтеров, зарабатывать на чем-нибудь свои деньги. Сделай они дешевле свои картриджи, им пришлось бы повысить цену на свои принтеры. Главное запомнить следующее: струйные принтеры обходятся дешево только тогда, когда на них мало печатают, а лазерные принтеры имеют более высокую стоимость изначально, но их расходные материалы стоят дешевле и служат дольше.

Полноцветная страница на струйном принтере может обходиться в 20–50 раз дороже аналогичной страницы на лазерном принтере, а также требовать специальной бумаги и распечатываться гораздо медленнее. Чернильные картриджи, используемые в струйных принтерах, быстро заканчиваются и часто засыхают или портятся. При намокании чернила растекаются, поэтому использовать струйный принтер для распечатки книги рецептов, которой потом будут пользоваться на кухне, не стоит. С другой стороны, на струйных принтерах сегодня можно получать фотоснимки, по качеству ничуть не уступающие тем, что печатаются в специальных фотолабораториях. На лазерных принтерах цветные фотоснимки получаются хуже.

Во всех принтерах есть подверженные неисправностям детали. Дешевые принтеры ломаются быстрее.

Другими словами, все это компромиссы. Для мелкосерийной печати в личных целях — печати одной или двух Web-страниц в день или печати двух фотопленок в месяц — прекрасно подойдет недорогой универсальный струйный принтер.

Когда в следующий раз пойдете покупать принтер, подумайте, как долго вы хотите, чтобы он прослужил, как часто вы что-нибудь печатаете и что именно вам обычно нужно распечатывать, и только потом покупайте. Посчитайте срок службы и стоимость страницы для каждого возможного кандидата и выберите наиболее оптимальный вариант. А также не забудьте узнать в своем локальном отделении компании по замене картриджей, берутся ли там заправлять картриджи для данного принтера и сколько это стоит.

Организуйте учет использования принтера

При установках среднего и большого размера рассматривайте как вариант организацию учета ресурсов принтера, даже если вы не собираетесь взимать плату за печать. На скорость распечатывания заданий это почти никак не влияет, зато позволяет в любой момент взять и посмотреть, кто и насколько часто пользуется принтером. Демографическая информация об источниках заданий на печать очень полезна при выборе мест для установки новых принтеров.

Для CUPS было разработано несколько программ учета ресурсов принтера (наиболее известными среди которых являются `accsnmp` и PyKota). На сайте компании ESP доступен централизованный, поддерживающий возможность поиска список ссылок на эти и другие имеющие отношение к CUPS-продукты: `www.easysw.com/~mike/cups/links.php`.

Защищайте принтеры

Сетевые принтеры, как правило, поддерживают возможность удаленного управления. Т.е. они позволяют администраторам выполнять их конфигурирование и мониторинг по

сети через протокол IPP или SNMP или из Web-браузера при помощи протокола HTTP. Через удаленный интерфейс администраторы могут устанавливать такие параметры, как IP-адрес принтера, стандартный шлюз, сервер Syslog, групповое имя (пароль) SNMP, опции протокола и пароль администратора.

По умолчанию почти все допускающие удаленное администрирование принтеры являются незащищенными и используют пароль (или "групповое имя" SNMP), который был назначен во время процесса установки. Информация о том, как именно это делается на том или ином принтере, должна быть доступна в прилагавшемся к этому принтеру руководстве по установке, но средства администрирования с графическим интерфейсом в CUPS и KDE Print Manager все чаще и чаще скрывают сведения о различиях производителей. И эта тенденция будет продолжаться.

23.8. Печать в KDE

Мы уже упоминали о KDE несколько раз ранее в этой главе [3]. Имеющиеся у KDE возможности для печати, однако, являются довольно удобными и потому заслуживают того, чтобы о них рассказали более подробно. KDE приложила немало усилий для того, чтобы ее средства и интерфейсы для печати не зависели от лежащей в основе системы печати. Она появилась уже после того, как CUPS стала популярной, и потому поддерживает все функциональные возможности CUPS. Работает она, однако, со всем, начиная от LPRng и заканчивая общей внешней программой.

Возможности печати в GNOME отстают от тех, что доступны в KDE, но разработчики GNOME хотят, чтобы их пользователи тоже получали хорошее впечатление от печати. Процесс разработки идет полным ходом; не исключено, что к моменту выхода этой книги возможности печати GNOME уже смогут конкурировать с аналогичными возможностями KDE. Один из читавших первый черновой вариант этой книги человек очень тонко подметил, что CUPS заменила "воюющие стандарты печати, у которых не было смысла жить, но которые отказывались умирать", только для того, чтобы вызвать конкуренцию между комплектами настольных утилит для печати.

Основной средой для печати в KDE является программа KDEPrint. Она включает средства для добавления принтеров, администрирования заданий на печать, перезапуска серверов печати и т.д. Да, CUPS позволяет делать это все тоже, но средства KDEPrint обладают здесь двумя преимуществами.

Во-первых, они имеют характерный для KDE внешний вид, что делает их более понятными пользователям KDE. Например, `kghostview` представляет собой ту же программу Ghostview, но в более соответствующей KDE обложке. (Не трудно заметить, что даже имена утилит KDE имеют определенный внешний вид. Просто кто-то недавно спросил у нас, является ли `ksh` приложением KDE.)

Во-вторых, KDEPrint не зависит от спулера. Если по какой-то причине вы не используете систему CUPS (или, даже еще хуже, вынуждены переключаться то на одну, то на другую систему печати), вы все равно можете применять KDEPrint для управления

[3] KDE — это набор стандартов библиотек и пользовательских интерфейсов для графических интерфейсов, выполняющихся под управлением X Window System, технологии, на которой основываются все графические интерфейсы Linux. KDE является альтернативой системы GNOME, которая используется по умолчанию в большинстве дистрибутивов. Несмотря на отличия, делать выбор между KDE и GNOME вовсе не обязательно. Более подробное описание GNOME и KDE приводилось в разделе 22.5.

процессом печати. Однако заранее предупреждаем, что CUPS является мощнее других систем печати, так что если вы перейдете на какую-нибудь альтернативную систему печати, не исключено, что некоторые из функциональных возможностей KDEPrint вам больше не будут доступны.

Зачем вам волноваться обо всех этих графических интерфейсах, если вы всегда выполняете все связанные с печатью задачи из интерпретатора команд? Ну, ваши пользователи вряд ли будут пользоваться интерфейсом интерпретатора команд, так что, скорее всего, вам все равно придется иметь определенное представление об интерфейсе KDE просто для того, чтобы поддерживать их.

Ниже перечислены основные компоненты KDEPrint, о которых следует знать.

- Утилита с GUI-интерфейсом `kprinter`, которая позволяет отправлять задания на печать;
- Мастер Add Printer (Добавление принтера), который автоматически обнаруживает сетевые принтеры (JetDirect, IPP и SMB) и некоторые принтеры, подключаемые локально. Мастер Add Printer также позволяет добавлять и конфигурировать принтеры, которые не удается обнаружить автоматически.
- Утилита Print Job Viewer (Средство просмотра заданий на печать), которая позволяет менять местами и отменять задания на печать, а также просматривать информацию об их состоянии.
- Руководство *KDEPrint Handbook*, в котором содержится необходимая справочная информация по данной системе. Оно доступно через панель KDE Help Center (Справочный центр KDE), но там его может быть трудно найти. Легче вызвать какую-нибудь программу вроде `kprinter` и щелкнуть на кнопке Help (Справка) или же выполнить команду `konqueror help:/kdeprint`. Кроме того, документация по KDEPrint еще также доступна по адресу `printing.kde.org`.
- Программа Print Manager (Диспетчер печати), которая является главным средством управления с графическим интерфейсом для системы печати. Ее тоже может быть немного трудно найти. Можно попробовать поискать ее в главном меню рабочего стола, хотя ее расположение в дереве меню в каждом дистрибутиве выглядит по-разному. Еще один вариант — это выполнить команду `kcmshell printmgr` или `konqueror print:/manager`.

Мастер Add Printer и программа Print Manager доступны через либо `kprinter`, либо KDE Print Manager (не говоря уже об URL-адресах `print:/manager` и `print:/printers` в Konqueror).

Информация о пользователях KDEPrint хранится в файлах `~/.kde`. Эти файлы имеют понятный для человека формат и их можно изменять через программу Print Manager. Если хотите, попробуйте отредактировать что-нибудь в них на свой страх и риск.

`kprinter`: печать документов

`kprinter` — это имеющая графический (GUI) интерфейс версия программы `lpr`. Но ею точно также можно пользоваться и из командной строки, а также даже отключать графический интерфейс; команда

```
$ kprinter --nodialog -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

эквивалентна команде

```
$ lpr -5 -P lj4600 riley.ps gillian.pdf zoe.prn
```

Ваши пользователи, скорее всего, захотят пользоваться графическим интерфейсом. Покажите им, как можно перетащить файлы из окна диспетчера файлов или с рабочего стола в диалоговое окно `kprinter` и затем распечатать их все сразу. Замените `lpr` на `kprinter` в появляющемся в их браузере окне печати, и у них будет окно печати с графическим интерфейсом. Научите их отмечать флажок Keep this dialog open after printing (Оставлять это диалоговое окно открытым после печати), и они даже не будут тратить время на повторный запуск этой программы каждый раз, когда им будет необходимо напечатать что-нибудь.

Обратите внимание на меню Print System currently in use (Используемая на текущий момент система печати), показывающее, что система KDEPrint сейчас не активна. Также обратите внимание на то, что когда принтер не доступен, `kprinter` предлагает выполнить печать в PDF или распечатать документ через факс. Дополнительные опции тоже заслуживают внимания: они позволяют ставить в очередь на печать свое резюме и указывать, что оно должно быть распечатано после того, как начальник уйдет домой.

Konqueror и печать

Многие Web-браузеры распознают ряд специальных URI-адресов, которые выступают в роли шлюзов к уникальным функциональным возможностям. Вы наверняка пробовали вводить `about:config` и `about:mozilla` в окне Firefox. Точно так же семейство URI-адресов `print:` является в Konqueror шлюзом к KDEPrint.

При вводе URI-адреса `print:/` отображается перечень всех возможностей, при вводе `print:/jobs` — перечень заданий на печать, а при вводе `print:/manager` — внутри Konqueror запускается программа Print Manager.

Обратите внимание на то, что CUPS здесь не используется, по крайней мере напрямую. Это все часть уровня KDEPrint.

23.9. Рекомендуемая литература

- Michael Sweet. *CUPS: Common UNIX Printing System*. Indianapolis, Indiana: Sams Publishing, 2001. Это авторитетное руководство по CUPS, можно сказать “библия”.
- Мы упоминали о сайте `linuxprinting.org` несколько раз в этой главе. Он представляет собой обширную коллекцию различных источников информации по печати в Linux и является прекрасным местом для поиска ответов на свои вопросы. На этом сайте также выложено замечательно учебное пособие по CUPS, в котором имеется раздел, посвященный выявлению и устранению проблем.
- Wikipedia и SUSE предоставляют качественное краткое описание CUPS:

  ```
  en.opensuse.org/SDB:CUPS_in_a_Nutshell
  en.wikipedia.org/wiki/Common_Unix_Printing_System
  ```

- На странице `cups.org/newsgroups.php` доступна неплохая коллекция касающихся CUPS групп новостей. Здесь можно задавать вопросы, но сначала следует попробовать разобраться во всем самому и только потом ставить вопросы, причем делать это вежливо.
- В состав KDE входят справочные страницы с описанием команд KDEPrint и руководство *KDEPrint Handbook*. Дополнительную информацию можно найти на сайте `printing.kde.org`. Все эти источники содержат полезные ссылки на другую

документацию. (Даже если у вас нет KDE, в документации по KDE вы сможете найти неплохую общую информацию о CUPS.)

23.10. Упражнения

23.1. При помощи браузера зайдите на сервер CUPS в своей сети. Что мешает вносить административные изменения в обслуживаемые этим принтером серверы?

23.2. Отыщите кого-нибудь, кто не силен в компьютерах (ученика школы искусств, свою маму или хотя бы дипломированного специалиста по программным средствам корпорации Microsoft) и научите этого человека распечатывать PDF-документ в системе Linux. Были ли какие-нибудь шаги не понятны вашему подопытному? Как можно было бы упростить этот процесс для других пользователей?

23.3. Посетите какой-нибудь настоящий или виртуальный гипермаркет типа Sam's Club или Amazon.com и составьте список принтеров, которые можно приобрести по цене ниже $250. Если бы вам пришлось покупать один из этих принтеров для своей организации завтра, какой из них вы бы выбрали и почему? Обоснуйте свой ответ при помощи данных из базы данных на `linuxprinting.com.org`.

23.4. Вас попросили спроектировать систему программного обеспечения для основанного на Linux лазерного принтера, ориентированного на рынок корпоративных рабочих групп. С какого дистрибутива Linux вы начнете? Какие дополнительные программы вы добавите, и какие программы вам придется написать самостоятельно? Как вы адаптируете Windows- и MacOS-клиентов? (Подсказка: ищите дистрибутивы Linux, предназначенные для "встраиваемых систем".)

Book_LinuxAdmHand.indb 868 17.04.2007 13:56:25

Глава 24

Обслуживание аппаратных средств и условия эксплуатации

Увеличение количества настольных рабочих станций и отход от “больших железных ящиков” на какое-то время заставили всех подумать, что дни центральных машинных залов (также известных под именем “центры обработки данных”) сочтены. Не тут-то было! За последние десять лет эти настольные системы стали очень сильно зависеть от центральных серверов, функционирующих под управлением операционных систем, подобных Linux. В результате в некогда покинутых машинных залах теперь часто можно увидеть ряды серверов.

Обеспечение благоприятных условий эксплуатации для этих серверов является не менее важной задачей. На самом деле требования по электропроводке и проветриванию у многих новейших устанавливаемых в стойку серверов 1U[1] часто соответствуют или даже превышают требования мэйнфреймов, которые они заменили.

В этой главе приводятся советы по обслуживанию аппаратных средств и поддержанию рабочих помещений в надлежащем состоянии.

[1] 1U — минимальная стандартная высота устройств, устанавливаемых в стойку. Равняется 1,75 дюйма, или 44,45 мм.

24.1. Основы обслуживания аппаратных средств

Техническое обслуживание аппаратных средств традиционно считалось областью, в которой заключались дорогостоящие ежегодные контракты. Хотя такие контракты до сих пор практикуются, сегодня более популярным и "дешевым" подходом является "взваливание этих обязанностей на плечи системного администратора".

Если в организации ведется регистрационная книга, то беглый просмотр записей за последние полгода-год даст представление об интенсивности сбоев аппаратуры. Рекомендуем тщательно вести список сбоев и замен комплектующих изделий, чтобы всегда можно было дать точную оценку различным вариантам обслуживания. Некоторые элементы сбоят чаще, чем предусматривается фирмой-изготовителем, поэтому контракты иногда не только удобны, но и выгодны с финансовой точки зрения. Однако не забывайте: приходит время, когда все оборудование нужно уже не обслуживать, а менять. Не пропустите момент и вовремя дайте своим аппаратным средствам "уйти на пенсию". Можно даже подарить списанное оборудование местной школе или университету. Для них оно редко бывает слишком старым.

О списании оборудования будет более подробно рассказываться в разделе 30.6.

Планируя стратегию технического обслуживания, подумайте, какие компоненты наиболее предрасположены к преждевременному износу. Устройства, имеющие сменные детали, как правило, являются менее надежными, чем "цельные" устройства вроде ЦП и ЗУ. Ниже перечислены наиболее вероятные кандидаты на быструю "отправку на пенсию".

- Лентопротяжные устройства.
- Устройства автоматической загрузки и замены ленты.
- Драйверы жестких дисков.
- Вентиляторы.
- Клавиатуры.
- Мыши.
- ЭЛТ-мониторы.

24.2. Контракты на обслуживание

Некоторые крупные компании предлагают услуги по обслуживанию аппаратных средств, выпускаемых другими фирмами. Эти компании часто стремятся вытеснить изготовителя и занять его место. Иногда можно добиться очень выгодного контракта на техническое обслуживание, настроив изготовителя против такого "посредника". Если есть возможность, соберите сведения обо всех потенциальных сервисных центрах.

Обслуживание на месте

Если контракт предусматривает обслуживание на месте, то специалист сервисного центра принесет запасные детали прямо в офис. Гарантированный срок его прихода колеблется от четырех до двадцати четырех часов и, как правило, оговаривается в контракте. В рабочее время вызванный специалист обычно приходит гораздо быстрее, чем в нерабочее.

Прежде чем заключать дорогостоящий контракт, предусматривающий максимальную оперативность обслуживания, подсчитайте, а не дешевле ли создать несколько полнофункциональных резервных систем на случай оперативной замены вышедших из строя компьютеров? Такую скорость замены не обеспечит ни один контракт, а с учетом падения цен на компьютерное оборудование стоимость инвестиций может оказаться действительно небольшой.

Обслуживание с заменой плат

Контракт на замену плат требует от администратора и его помощников самостоятельной диагностики проблем, возможно, при помощи специалистов службы круглосуточной поддержки фирмы-изготовителя. После выявления проблемы нужно позвонить в службу, описать проблему и заказать необходимую плату. Плата, как правило, отправляется немедленно и поступает на следующий день. Нужно установить ее, включить компьютер, убедиться, что все работает нормально, и вернуть на фирму старую плату в той же упаковке, в которой была получена новая.

Обычно изготовитель присваивает каждой сервисной операции номер разрешения на возврат покупки. Его необходимо написать на сопроводительных документах, отправляемых со старой платой.

Гарантийные обязательства

Гарантийный срок, установленный изготовителем устройства, нужно обязательно учитывать при расчете затрат на эксплуатацию оборудования. В большинстве случаев лучше всего придерживаться стратегии “избирательной гарантии”. Изготовители дисководов дают гарантию до пяти лет, а у некоторых модулей памяти срок работы вообще не ограничен. На компьютеры обычно предоставляется годичная гарантия, но бывают и более выгодные условия. Покупая новую технику, выбирайте ту, у которой наибольший гарантийный срок, — это позволит в конечном итоге сэкономить деньги.

Во многих организациях, похоже, гораздо легче “выбить” средства на покупку капитального оборудования, чем на содержание обслуживающего персонала и оплату сервисных услуг. Мы иногда оплачивали “продленную гарантию” на новое оборудование (ее можно считать также обслуживанием с предоплатой), чтобы превратить доллары, выделенные на закупку оборудования, в доллары, которые можно потратить на техническое обслуживание.

Если оборудование состоит из множества компонентов, проблемы с обслуживанием и надежностью обычно начинают возникать почти сразу же после их установки. Отказы оборудования, которые происходят в течение дня или двух с момента его установки, называются “выходом из строя в начальный период эксплуатации” (infant mortality).

24.3. Уход за электронным оборудованием

С печатными платами и другими электронными устройствами следует обращаться аккуратно, не ронять их, не проливать на них кофе, не класть на них книги, и т.п. Многие наладчики (имеются в виду обаятельные специалисты, присылаемые фирмой, с которой ваша организация заключила контракт на техническое обслуживание) нередко обращаются с оборудованием раз в десять грубее положенного.

Статическое электричество

Электронные компоненты чувствительны к статическому электричеству. Чтобы находиться в безопасности при работе с ними, перед и во время их установки следует заземляться. Антистатический браслет, одетый на запястье и подсоединенный к "заземлению" (обычно доступному в виде третьего штырька в розетке), надежно защитит вас.

Помните о том, что беспокоиться о статическом электричестве нужно и при первом вскрытии упаковки с электронным компонентом, и при любом другом последующем контакте с ним, а не только во время его установки. Будьте особенно осторожны, если пол в помещении офиса, куда вам доставляют почту (и где вы, следовательно, вскрываете все пакеты) покрыт ковром: ковры генерируют больше статического электричества, чем обычные полы.

Вот один из способов снятия заряда с ковровых покрытий. Купите в отделе хозяйственных товаров смягчитель для тканей и раз в месяц распыляйте 30-процентный водный раствор этого средства над ковром (на компьютеры брызгать не надо). Эта процедура позволяет существенно снизить уровень статического заряда, и, кроме того, в помещении будет стоять свежий апрельский аромат.

Переустановка плат

Иногда аппаратную проблему можно устранить, выключив питание, переустановив интерфейсные платы (SCSI, Ethernet и т.д.) и включив питание вновь. Если проблема на некоторое время исчезает, но через неделю или месяц возникает вновь, это, скорее всего, свидетельствует о плохом контакте между интерфейсной и материнской платами.

Контакты можно прочищать специальным моющим раствором или же обычной карандашной резинкой. Резинка не должна быть старой и жесткой; если она плохо стирает карандашные линии с бумаги, то электрические контакты она тем более не очистит. Старайтесь не касаться контактов пальцами. Протрите их резинкой, отряхните крошки и установите плату на место.

На некоторых материнских платах до сих пор есть микросхемы в гнездах. Со временем качество соединений в гнездах ухудшается (главным образом из-за вибрации, вызванной работой вентиляторов). Попробуйте восстановить контакт, сильно нажав на микросхему большим пальцем (естественно, сначала следует надеть антистатический браслет).

24.4. Мониторы

За последние несколько лет цены на ЖК-мониторы к счастью снизились на столько, что теперь они могут устанавливаться повсеместно. Хотя первоначальная стоимость ЖК-мониторов немного выше стоимости ЭЛТ-мониторов, эти устройства потребляют меньше энергии, требуют меньшего ухода и обычно не так сильно напрягают глаза, как их основанные на ЭЛТ предшественники. Если в вашей организации до сих пор используются ЭЛТ-мониторы, наилучшим планом по обслуживанию будет просто заменить их на ЖК-мониторы.

Если заменить ЭЛТ-мониторы ЖК-мониторами не представляется возможным, тогда знайте, что во многих моделях ЭЛТ-мониторов важные регуляторы яркости и сведения лучей находятся непосредственно на плате, скрытой под корпусом. К сожалению, напряжение внутри ЭЛТ-мониторов часто достигает десятков тысяч вольт и сохраняется

на протяжении еще очень длительного времени даже после отключения питания. Из-за опасности поражения электрическим током мы рекомендуем для настройки ЭЛТ-монитора всегда приглашать квалифицированного специалиста. Не пытайтесь делать это самостоятельно.

24.5. Модули памяти

В современных аппаратных средствах в основном применяются не отдельные микросхемы памяти, а модули SIMM (Single In-line Memory Module — модуль памяти с однорядным расположением выводов), DIMM (Dual In-line Memory Module — модуль памяти с двухрядным расположением выводов) или RIMM (Rambus Inline Memory Module — модуль памяти Rambus). Емкость этих модулей, устанавливаемых на одной маленькой плате, составляет от 32 Мбайт до 4 Гбайт.

Если нужно расширить память рабочей станции или сервера, можно заказать модули памяти в какой-нибудь третьей фирме и установить их самостоятельно. Старайтесь не покупать модули памяти у производителей компьютеров; они, как правило, сильно завышают цену[2]. Добавляя память, будьте щедрыми. Цены на память постоянно снижаются, одновременно с этим уменьшается и число слотов расширения на типичной материнской плате.

Перед покупкой модулей памяти тщательно проверьте в документации к компьютеру, какие типы модулей поддерживаются. Часто можно увеличить производительность системы за счет установки модулей с повышенной частотой работы или со специальными возможностями, такими как DDR (Double Data Rate — двойная скорость передачи данных). Уточните количество свободных слотов и проверьте, нет ли каких-либо ограничений на добавление новых модулей. В некоторых материнских платах модули должны добавляться парами. Иногда этого не требуется, однако производительность повышается от парного добавления модулей.

Проверьте, могут ли старые и новые модули эффективно взаимодействовать друг с другом. В большинстве случаев задействуются только возможности, общие для всех модулей.

При самостоятельной установке модулей памяти помните о том, что модули памяти в большей степени, чем другие устройства, чувствительны к электростатическим зарядам. Перед тем как открывать упаковку с модулями памяти, нужно убедиться в наличии заземления.

Контакты модулей памяти тоже допускается чистить карандашной резинкой, как описывалось выше.

24.6. Профилактическое обслуживание

Может показаться странным (ведь некоторые из нас думают, что мы уже избавились от этой проблемы), но во многих устройствах имеются воздушные фильтры, которые нужно обязательно регулярно чистить или заменять. Засоренные фильтры препятствуют циркуляции воздушного потока, что может приводить к перегреву, а это одна из основных причин аппаратных сбоев. Вентиляционные отверстия следует держать открытыми на всех устройствах, но особенно внимательно нужно относиться к тем серверам, кото-

[2] Если они предлагают модуль памяти в виде части пакета, тогда другое дело: иногда такие варианты оказываются вполне приемлемыми.

рые теснятся в маленьких стойках размера 1U или 2U. Охлаждение этих систем зависит от вентиляции. Без нее расплавления основных элементов не избежать.

Все устройства с движущимися механизмами требуют регулярной смазки, чистки и профилактики приводных узлов. Следите за старым оборудованием и, едва заслышав скрип, принимайте соответствующие меры.

В серверных системах чаще всего ломаются вентилятор и блок питания. Это особенно актуально для персональных компьютеров, где оба устройства заключены в один сменный модуль. Периодически проверяйте функционирование вентиляторов. Если они начинают выходить из строя, необходимо заменить весь модуль, так как возможен перегрев компьютера. Не пытайтесь самостоятельно смазывать вентилятор: это может отсрочить неизбежный крах, а может и усугубить проблему либо привести к повреждению других компонентов.

Во многих персональных компьютерах имеется отсек для второго вентилятора (и разъемы питания к нему). Если уровень шума не имеет значения, всегда рекомендуется устанавливать второй вентилятор[3]. Он не только снижает рабочую температуру, но и служит резервом на случай выхода из строя основного вентилятора. Вентиляторы дешевы, поэтому всегда можно держать в запасе несколько штук.

В пыльной среде компьютерные компоненты выходят из строя гораздо чаще, чем в относительно чистом помещении. Пыль набивается в воздушные фильтры, высушивает смазку, засоряет движущиеся механизмы (например, вентиляторы) и покрывает устройства слоем "изоляции", снижающим их способность рассеивать тепло. Все это приводит к повышению рабочей температуры. При работе в неблагоприятных условиях рекомендуется время от времени проводить внутреннюю чистку компьютера (любое помещение, в котором есть ковер, является неблагоприятной средой).

Удалять пыль лучше всего специальными пылесосами, но держите мотор не менее чем в полутора метрах от системных компонентов и жестких дисков для уменьшения влияния электромагнитных полей. В компьютерном помещении следует регулярно проводить уборку, но это должен делать обученный человек, умеющий соблюдать дистанцию и беречь оборудование от повреждений (уборщицам, как правило, доверять нельзя).

Лентопротяжные устройства тоже требуют регулярной чистки. Большинство накопителей чистится с помощью специальной чистящей кассеты.

24.7. Условия эксплуатации

Подобно людям, компьютеры работают лучше и дольше, если находятся в благоприятной среде. Конечно, им не важно, какой вид открывается из окна, но за другими параметрами помещения необходимо внимательно следить.

Температура

Идеальная рабочая температура для компьютерной техники — от 17 до 20°C при влажности около 45%. К сожалению, эти показатели не совпадают с идеальной рабочей температурой для пользователей. При температуре в компьютерном помещении выше 27°C внутри компьютеров создается температура около 49°C. Для серийно выпускаемых микро-

[3] Если уровень шума имеет значение, можно попробовать установить какой-нибудь из новейших "сверхтихих" вентиляторов. Более подробная информация доступна на сайте `www.silentpcreview.com`.

схем верхняя граница диапазона рабочих температур составляет как раз 49°C (в этой точке они перестают работать), а при температуре свыше 71°C микросхемы выходят из строя. Температура входного воздуха тоже играет очень важную роль: горячие выхлопы одной машины никогда не должны попадать в воздухозаборное устройство другой машины.

Влажность

Идеальная влажность для большинства компьютерных устройств — от 40 до 55%. Если влажность слишком низкая, возникают проблемы, связанные с электростатическими зарядами. Если влажность, наоборот, слишком высокая, влага конденсируется на платах, вызывая короткие замыкания и окисление.

Охлаждение офисных помещений

В наши дни многие компьютеры эксплуатируются в офисах и вынуждены "выживать" в условиях общеобменной системы вентиляции (которая часто выключается по ночам и на выходные), с кипами газет и книг на вентиляционных отверстиях. Устанавливая компьютер в офисе, помните, что он будет поглощать воздух, которым дышат люди. При расчете требуемой производительности системы охлаждения следует соблюдать правило: каждый человек в помещении вырабатывает 300 тепловых единиц (БТЕ) в час (примерно 0,1 кВт), тогда как средний офисный персональный компьютер — 1100 единиц (примерно 0,3 кВт). Не забывайте также о тепловой нагрузке, создаваемой окнами, которые пропускают прямой солнечный свет.

Охлаждение машинных залов

Главной заботой тех, кому повезет разместить свои серверы в одном из тех модных машинных залов с фальшполом, которые были построены в 80-е гг. и оснащены средствами, способными охладить всю технику в зале и даже целый штат, скорее всего, будет то, где и как получить знания, необходимые для того, чтобы обслуживать такую примитивную систему кондиционирования. Остальным из нас придется научиться правильно рассчитывать производительность охлаждающей подсистемы. Хорошо охлаждаемый машинный зал — это успешный машинный зал.

Мы обнаружили, что расчеты тепловой нагрузки, выполняемые специалистами из фирмы по установке кондиционеров, лучше перепроверять, особенно когда речь идет об установке системы для машинного зала. В чем им можно доверять полностью, так это в расчете тепловой составляющей (теплопритока), приходящейся на потолок, стены и окна (не забывайте также о прямом солнечном свете). В данной области у инженеров по системам отопления, вентиляции и кондиционирования воздуха накоплен большой опыт и их оценки достаточно точны. Проверки требует только расчет внутренней тепловой нагрузки компьютерного помещения.

Свой вклад в общую оценку вносят следующие компоненты:

- потолок, стены и окна (расчет предоставляется специалистом);
- электронные устройства;
- осветительные приборы;
- операторы (люди).

Электронные устройства

Для вычисления количества тепла, выделяемого серверами (и другими электронными устройствами), необходимо определить их потребляемую мощность. Проще всего снять показания непосредственно со счетчика электроэнергии. Можно попросить помочь с этим своего знакомого соседа-электрика или приобрести недорогой счетчик и сделать это самостоятельно[4]. На устройства обычно имеются паспорта, где указана максимальная потребляемая мощность в ваттах, но в нормальных условиях потребление энергии не достигает максимума. Тепловая мощность измеряется в других единицах — БТЕ/час (БТЕ — британская тепловая единица), поэтому значение потребляемой мощности необходимо умножить на 3,412 БТЕ/(ч·Вт). Например, если оборудуется компьютерное помещение с 25 серверами, каждый из которых потребляет 450 Вт, расчет будет таким:

$$(25\ \text{серверов})\left(\frac{450\ \text{Вт}}{\text{сервер}}\right)\left(\frac{3{,}412\ \text{БТЕ}}{\text{ч}\cdot\text{Вт}}\right) = 38385\ \text{БТЕ/ч}$$

Осветительные приборы

Как и в случае электронных устройств, вычисление тепловой нагрузки осветительных приборов основано на их потребляемой мощности. Типичный офисный светильник содержит четыре 40-ваттные флуоресцентные трубки. Если в новом помещении шесть таких светильников, получится следующий расчет:

$$(6\ \text{светильников})\left(\frac{160\ \text{Вт}}{\text{светильник}}\right)\left(\frac{3{,}412\ \text{БТЕ}}{\text{ч}\cdot\text{Вт}}\right) = 3276\ \text{БТЕ/ч}$$

Операторы

В компьютерном помещении регулярно работают люди. Тепловая нагрузка от каждого из них — 300 БТЕ/ч. Предположим, одновременно в помещении находятся четыре человека. Тогда:

$$4 \times 300\ \text{БТЕ/ч} = 1200\ \text{БТЕ/ч}$$

Общая тепловая нагрузка

После вычисления тепловой нагрузки каждого компонента необходимо сложить эти показатели и получить общую тепловую нагрузку. Допустим, специалист по установке кондиционеров подсчитал, что тепловая составляющая потолка, стен и окон равна 20 000 БТЕ/ч. Тогда:

```
 20000   БТЕ/ч (потолок, стены, окна)
 38385   БТЕ/ч (серверы и другие электронные устройства)
  3276   БТЕ/ч (осветительные приборы)
  1200   БТЕ/ч (операторы)
-----------------------------------------------------------
 62861   БТЕ/ч (всего)
```

[4] Наиболее популярными счетчиками сегодня являются киловаттные счетчики KILL A WATT, которые выпускает компания P3 и которые стоят в районе $30.

Производительность системы охлаждения обычно выражается в холодильных тоннах. Одна холодильная тонна — это 12000 БТЕ/ч. Необходимо также ввести коэффициент ухудшения (не менее 50%), чтобы учесть погрешность вычислений и будущий рост тепловой нагрузки:

$$\left(\frac{62681\ \text{БТЕ}}{\text{ч}}\right)\left(\frac{1\ \text{т}\cdot\text{ч}}{12000\ \text{БТЕ}}\right)\times 1{,}5 = 7{,}84\ \text{холодильных тонн}$$

Проверьте, то ли это значение, которое вам сообщил специалист.

Контроль температуры

При эксплуатации компьютеров в промышленных условиях рекомендуется вести контроль температуры (а также других факторов окружающей среды, например шума и потребляемой мощности) в помещении, даже когда администратора нет на месте. Никому не хочется после прекрасно проведенных выходных прийти в понедельник на работу и обнаружить на полу лужицу расплавленного пластика. К счастью, есть системы автоматического контроля компьютерных помещений. Мы рекомендуем семейство продуктов Sensaphone компании Phonetics. Это недорогие устройства, позволяющие следить за параметрами окружающей среды, такими как температура, шум и потребляемая мощность, и сообщать о возникших проблемах звонком на телефон (или пейджер) администратора. Узнать об этих устройствах можно на Web-сайте `www.sensaphone.com`.

24.8. Источники питания

Компьютеры любят хорошее, стабильное и "экологически чистое" питание. Для компьютерного помещения это означает наличие стабилизатора напряжения — дорогого устройства, которое фильтрует скачки напряжения и позволяет устанавливать нужные уровни и фазы. В офисах можно использовать сетевые фильтры, которые помогут уберечь аппаратуру от перепадов в сети.

Серверы и сетевое оборудование нужно включать в электросеть через источник бесперебойного питания (Uninterruptible Power Supply, UPS), или ИБП. Хороший ИБП может подключаться к компьютеру через интерфейс RS-232, Ethernet или USB. Это позволяет ИБП предупреждать компьютер о том, что напряжение в сети пропало и что он должен успеть выключиться до того, как разрядится аккумулятор.

Проведенное исследование показало, что 13% электроэнергии, потребляемой в США, используется для питания компьютеров. Традиционно UNIX-системы строились на базе аппаратных и программных средств, которые были рассчитаны на круглосуточную работу. Сегодня 24 часа в сутки должны работать только серверы и некоторые сетевые устройства. Настольные компьютеры могут выключаться вечером самими пользователями (если те умеют делать это корректно).

📖 Подробнее о процедуре останова системы рассказывалось в разделе 2.6.

Бывают случаи, когда приходится регулярно перезагружать сервер из-за аппаратной проблемы или сбоев ядра. Возможно также, что в помещении имеются серверы Windows, которые сильнее подвержены подобным сбоям. Так или иначе, может потребоваться установить систему, которая позволит осуществлять дистанционную перезагрузку серверов.

Наиболее приемлемое решение предлагает компания APC. Ее продукт MasterSwitch может управляться посредством Web-браузера через встроенный Ethernet-порт. Получить информацию об этом продукте можно на Web-сайте `www.apcc.com`.

24.9. Стойки для аппаратуры

Эпоха машинных залов с настоящими фальшполами, в которых все кабели электропитания, системы охлаждения, сети и телефонных соединений скрывались под полом, прошла. Вы когда-нибудь пробовали проследить проводку кабелей в одном из таких проходящих под полом лабиринтов? Наш опыт показывает, что "классический" зал с фальшполом, хотя и выглядит хорошо через стекло, на самом деле является скрытым крысиным гнездом. Сегодня фальшпол можно использовать только для сокрытия электропроводки, распределения охлажденного воздуха и *все*! Ни для чего другого его использовать не рекомендуется.

При обустройстве компьютерных помещений профессионального уровня важно выделить комнату для компьютеров серверного класса. В этой комнате можно регулировать температуру и другие параметры окружающей среды; она также отвечает требованиям физической безопасности серверов.

С профессиональной точки зрения в специализированных компьютерных помещениях оборудование должно храниться в стойках (а не, скажем, на столах или на полу). В лучших современных системах стойки соединены подвесными конструкциями для прокладки кабелей. Это придает помещению высокотехнологичный вид, не усложняя его обслуживание.

Самую лучшую сборочную систему производит компания Chatsworth Products (Четсворт, шт. Калифорния, тел.: (818) 882-8595). Используя 19-дюймовые (19 дюймов = 48,26 см) монорельсовые телекоммуникационные стойки можно построить "шкаф" как для размещаемых на полках, так и для устанавливаемых в стойках серверов. Две стоящих спина к спине 19-дюймовые стойки дают одну имеющую высокотехнологичный вид "традиционную" стойку (удобную в случаях, когда детали стоек должны крепиться как спереди, так и сзади оборудования). Chatsworth Products предлагает стойки, кабельные полки и детали для распределения кабелей, а также все необходимое для их сборки оборудование. Поскольку кабели размещаются на видимых дорожках, их легче отслеживать и, соответственно, содержать в чистоте.

24.10. Стандарты для центров по обработке данных

Серверные залы стали настолько частым явлением, что появились стандарты по их обустройству. Эти стандарты обычно описывают такие параметры, как разнообразие внешних сетевых соединений, доступные системы охлаждения и электроснабжения (вместе с их резервными подсистемами) и ежегодный период простоя по причине техобслуживания. Один такой список стандартов был опубликован институтом Uptime Institute; вошедшие в него основные категории вкратце приведены в табл. 24.1.

Помимо подробного описания каждого из этих уровней и того, как их можно достигнуть, институт Uptime Institute также предоставляет статистические данные и рекомендации по самым разным темам, касающимся инфраструктуры отказоустойчивых центров по обработке данных, которые доступны на его сайте `www.upsite.com`.

Таблица 24.1. Стандарты для серверов, предложенные институтом Uptime Institute

Уровень	Период безотказной работы	Питание/охлаждение	Избыточные компоненты
I	99.671%	Один канал	Избыточных компонентов нет
II	99.741%	Один канал	Несколько избыточных компонентов
III	99.982%	Много каналов. Активен только какой-нибудь один из них.	Избыточные компоненты, параллельное обслуживание
IV	99.995%	Много каналов. Активен более, чем один из них.	Избыточные компоненты, полностью отказоустойчивые

24.11. Инструменты

Чем лучше оснащен системный администратор, тем эффективнее он работает. Важно всегда иметь под рукой набор необходимых инструментов, чтобы быстро реагировать на возникающие проблемы. В табл. 24.2 приведен перечень инструментов, которые администратор должен держать под рукой или хранить в легкодоступном месте.

Таблица 24.2. Инструментарий системного администратора

Инструменты общего назначения	
Набор крестообразных отверток	Пинцет
Набор плоских отверток	Ножницы
Электротехнический нож	Набор торцевых ключей с удлиненными головками
Плоскогубцы и круглогубцы	Фонарик
Мелкие ювелирные отвертки	Набор шестигранных ключей
Молоток с круглым бойком	Набор ключей типа TORX (шестиугольная звезда)
Специальные компьютерные инструменты	
Устройство для зачистки проводов (со встроенными кусачками)	Переносной сетевой анализатор
Кабельные соединители	Запасной сетевой шнур
Запасные переходные кабели категории 5 для разъемов RJ-45	Обжимной инструмент для разъемов RJ-45
Запасные разъемы RJ-45	Терминаторы SCSI
Цифровой мультиметр	Коммутационный тестер
Антистатический браслет	
Разное	
Список адресов сервисных центров[а]	Ватные палочки
Номера телефонов и пейджеров специалистов, работающих по вызову	Изолента
Аптечка	Зубоврачебное зеркальце
Шесть упаковок хорошего пива[б]	Мобильный телефон

[а] И номеров гарантийных талонов, если они есть.

[б] Рекомендуемый минимум.

24.12. Рекомендуемая литература

Ниже перечислены источники, в которых можно найти дополнительную информацию о стандартах для центров по обработке данных.

- *Telecommunications Infrastructure Standard for Data Centers*. ANSI/TIA/EIA 942.
- ASHRAE INC. *ASHRAE Thermal Guidelines for Data Processing Environments*. Atlanta, GA: ASHRAE, Inc., 2004.
- Eubank, Huston, Joel Swisher, Cameron Burns, Jen Seal и Ben Emerson. D*esign Recommendations for High Performance Data Centers*. Snowmass, CO: Rocky Mountain Institute, 2003.

24.13. Упражнения

24.1. Зачем компьютеры иногда располагают в стойках?

☆ 24.2. Факторы окружающей среды влияют как на людей, так и на технику. Дополните факторы, перечисленные в этой главе, другими, актуальными для вас (например, пыль, шум, яркий свет, суматоха и т.д.). На примере любых четырех факторов покажите, пригодно ли ваше помещение для работы людей и компьютеров.

☆ 24.3. Сила тока в рабочей станции равна 0,8 А, а в мониторе — 0,7 А при напряжении 120 В.

а) Какова потребляемая этой системой мощность в ваттах? (Подсказка: $P = IU$.)

б) Если стоимость электроэнергии составляет 0,1$ за 1 кВт·ч, то во сколько обойдется непрерывная работа системы на протяжении года?

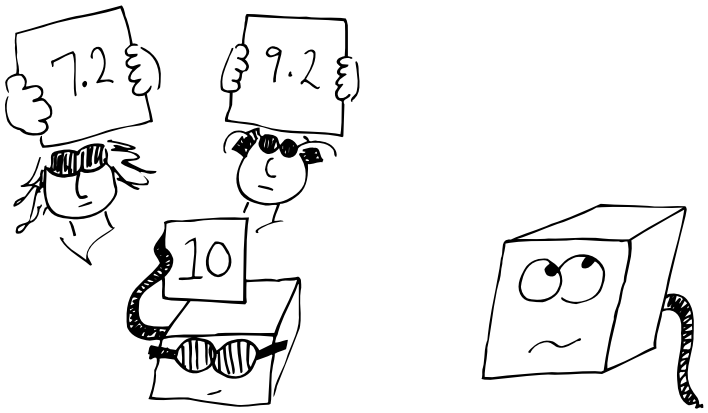
в) Сколько денег можно сэкономить за год, если выключать монитор каждый день в среднем на 16 часов (вручную либо средствами стандарта Energy Star)?

г) Какова годовая стоимость охлаждения системы? (Узнайте тарифы на эксплуатацию систем охлаждения и покажите расчеты.)

☆☆ 24.4. Составьте проект новой компьютерной лаборатории для вашей организации. Укажите предполагаемую площадь помещения, количество компьютеров, тип каждого компьютера и потребляемую им мощность. Вычислите суммарную мощность и тепловую нагрузку лаборатории. Включите в схему как серверы, так и рабочие станции. Учитывайте план помещения, расположение источников света и предполагаемое количество людей, работающих в помещении.

Глава 25

Анализ производительности



В этой главе мы сосредоточим внимание на производительности серверных систем. В настольных системах проблемы, характерные для серверов, обычно не возникают, поэтому ответ на вопрос о том, как повысить производительность настольной системы, обычно прост: “Провести модернизацию”. Пользователям такой ответ нравится, поскольку это означает, что у них появятся новые модные системы.

Одним из отличающих Linux от других основных операционных систем аспектов является объем данных, которые она предоставляет о своих внутренних операциях. Детальная информация доступна буквально по каждому уровню системы, благодаря чему администраторы могут настраивать различные параметры именно таким, каким нужно, образом. Если установить источник проблем с производительностью все никак не удается, всегда можно просмотреть исходный код. По этим причинам Linux часто считается наиболее подходящей операционной системой для пользователей, переживающих по поводу производительности.

К сожалению, настройка производительности в Linux не всегда является простой задачей. Как пользователи, так и администраторы часто думают, что если бы они только владели нужными “магическими” приемами, их системы работали бы в два раза быстрее. Одним из наиболее распространенных заблуждений является убеждение в том, что настроить производительность может помочь манипулирование переменными ядра, которые отвечают за управление системой страничного обмена и буферными пулами. В наши дни ядра основных дистрибутивов изначально настроены на достижение разумной (хотя и не оптимальной) производительности при различных уровнях загруженности. Если попытаться оптимизировать систему по какому-то конкретному показателю производительности (например, по степени использования буферов), весьма вероятно, что поведение системы ухудшится в отношении других показателей и режимов работы.

Источником самых серьезных проблем с производительностью обычно являются приложения, а не лежащая в основе операционная система. Если упомянутые приложения разрабатывали не вы сами у себя дома, тогда вам не повезло, потому что придется потратить массу времени на разговоры с представителями из команды поддержки их производителя. В этой главе будет рассказываться о том, как можно настроить производительность на уровне системы; возможность рассказать о настройке производительности на уровне приложений мы оставили другим.

В любом случае, никогда не доверяйте полностью тому, что пишут в Internet. Какие только аргументы не приводятся в дискуссиях по производительности систем! При этом сторонники всевозможных теорий не обладают, как правило, ни знаниями, ни дисциплиной, ни временем, необходимыми для проведения достоверных экспериментов. Широкая поддержка пользователей еще ничего не значит, ибо каждое глупое предложение очень часто сопровождается хором восторженных комментариев: “Я увеличил размер кэш-буфера в десять раз, как он предложил, и система заработала ГОРАЗДО, ГОРАЗДО быстрее!!!”. Да уж, конечно!

Настройка производительности — это нелегкая задача, которая требует терпения, методичного подхода и тщательного анализа. Ниже перечислены некоторые правила, которыми следует пользоваться.

- Собирайте и анализируйте хронологические данные о работе системы. Если еще неделю назад производительность системы была нормальной, а сейчас все изменилось, то сравнение характеристик системы в двух состояниях позволит выявить источник проблемы. Всегда держите под рукой список оптимальных параметров производительности. А также первым делом просматривайте журнальные файлы, чтобы выяснить, не связана ли возникшая проблема с появлением какой-нибудь неполадки в оборудовании.
- В главе 19 рассказывалось о некоторых средствах трендового анализа, которые также подходят для мониторинга производительности. Утилита `sar`, которая будет описываться чуть позже в этой главе, тоже может использоваться в качестве средства трендового анализа.
- Всегда настраивайте систему так, чтобы потом иметь возможность сравнить результаты с предыдущими показателями производительности системы.
- Всегда проверяйте наличие плана отката на случай, если внесенные “чудодейственные” изменения окажутся вовсе не чудодейственными и лишь только ухудшат дело.
- Не перегружайте умышленно свои системы или свою сеть. Linux создает для каждого процесса иллюзию бесконечности ресурсов. Но как только в дело вступают все 100% ресурсов системы, операционной системе приходится работать очень напряженно. На поддержку иллюзии расходуется ощутимая доля самих ресурсов. В результате выполнение процессов замедляется.

25.1. Способы повышения производительности

Ниже перечислены конкретные действия, которые можно выполнить, чтобы улучшить производительность.

- Убедиться в том, что память доступна в достаточном объеме. В следующем разделе вы увидите, что объем памяти очень сильно влияет на производительность.

Память сегодня стоит довольно недорого, так чтобы обычно оснастить любой чувствительный к производительность компьютер по максимуму — не проблема.

- Тщательно проверить конфигурацию системы и отдельных приложений. Многие приложения могут резко увеличить производительность, если их правильно настроить (распределить данные между дисками, не осуществлять динамический поиск имен в DNS, запустить дополнительные экземпляры популярного сервера и т.д.).
- Устранить проблемы, связанные с использованием ресурсов. Проблемы могут создаваться как пользователями (одновременный запуск слишком большого количества заданий, неэффективные методы программирования, выполнение заданий с завышенным приоритетом, запуск ресурсоемких программ в неподходящее время), так и самой системой (квоты, учет использования центрального процессора, ненужные демоны).
- Если Linux-система используется в качестве Web-сервера или сетевого сервера приложений, можно попробовать распределить трафик между несколькими компьютерами с помощью какого-нибудь коммерческого приложения для балансирования нагрузки, такого как Content Services Switch производства компании Cisco (`www.cisco.com`), ServerIron производства компании Foundry (`www.foundry.com`) или Alteon Web Switch производства компании Nortel (`www.nortelnetworks.com`)[1]. Эти устройства делают так, что несколько физических серверов выглядят для внешнего мира как один логический сервер. Они распределяют нагрузку согласно одному из нескольких выбираемых пользователем алгоритмов типа “минимальное время отклика” или “циклический алгоритм обслуживания”.
- Эти распределяющие нагрузку приложения также обеспечивают избыточность данных на случай, если один из серверов выйдет из строя. Они совершенно необходимы, если система должна выдерживать внезапные “всплески” активности.
- Организовать жесткие диски и файловые системы так, чтобы нагрузка на них была равномерной, и тем самым максимально повысить пропускную способность каналов ввода-вывода. При этом нужно удостовериться в том, что для диска выбран самый подходящий из доступных в Linux планировщик ввода-вывода. При наличии специфических приложений, таких как базы данных, можно попробовать использовать какую-нибудь модную многодисковую технологию вроде RAID для оптимизации процессов обмена данными. За рекомендациями следует обращаться к производителю базы данных.
- Обратить внимание на то, что разные приложения и СУБД ведут себя по-разному при размещении на нескольких дисках. Технология RAID поставляется в нескольких вариантах, поэтому следует потратить время и выяснить, какой из них (если хоть какой-нибудь) наиболее подходит для данного конкретного случая.
- Провести мониторинг сети, чтобы убедиться в том, что она не перегружена трафиком и что количество ошибок в ней минимально. Очень много такой полезной информации о сети можно собрать с помощью команды `netstat`, которая описывалась в разделе 19.4. Также можно еще заглянуть в главу 19.
- Определить ситуации, в которых система совершенно не удовлетворяет предъявляемым к ней требованиям.

[1] Бесплатной, хотя и менее стабильной альтернативой является приложение Linux Virtual Server, которое можно загрузить с сайта `www.linuxvirtualserver.org`.

Эти меры перечислены в порядке убывания эффективности. Добавление памяти и распределение трафика между несколькими серверами может значительно повысить производительность. Степень эффективности остальных мер варьируется от значительной до нулевой.

Анализ и оптимизация структур данных и алгоритмов программ почти всегда ведет к существенному повышению производительности. Однако эти вопросы обычно находятся вне компетенции системного администратора.

25.2. Факторы, влияющие на производительность

Производительность системы во многом определяется эффективностью распределения и совместного использования ее ресурсов. Определение термина "ресурс" весьма расплывчато. Оно может подразумевать даже такие компоненты, как кэш содержимого регистров центрального процессора и элементы таблицы адресов контроллера памяти. Однако в первом приближении серьезное влияние на производительность оказывают только четыре фактора:

- время использования центрального процессора;
- память;
- пропускная способность дисковой подсистемы ввода-вывода;
- пропускная способность сетевой подсистемы ввода-вывода.

Каждый процесс потребляет некоторую часть ресурсов системы. Если после того как активные процессы взяли все, что им нужно, остались свободные ресурсы, можно считать, что производительность системы является удовлетворительной.

Когда ресурсов не хватает, процессы становятся в очередь. Процесс, не имеющий немедленного доступа к необходимым ресурсам, должен ждать, ничего при этом не делая. Время, затрачиваемое на ожидание ресурсов, — один из основных показателей ухудшения производительности.

Самый простой с точки зрения учета ресурс — время использования центрального процессора (ЦП). В распоряжении системы всегда имеется примерно одна и та же часть его мощности. Теоретически эта величина составляет все 100% циклов ЦП, но "накладные расходы" и другие причины приводят к снижению этого показателя примерно до 95%. Для процесса, занимающего более 90% времени ЦП, ограничивающим фактором является быстродействие центрального процессора. Такой процесс потребляет большую часть вычислительных мощностей системы.

Многие считают, что быстродействие ЦП — основной фактор, влияющий на общую производительность системы. При наличии неограниченных объемов всех остальных ресурсов или для определенных типов приложений (например, программ численного моделирования) быстродействие ЦП *действительно* играет роль. Но в повседневной жизни этот показатель значит относительно мало.

Настоящим узким местом в Linux-системах является канал обмена данными с жестким диском. Жесткие диски представляют собой механические системы, поэтому на поиск нужного блока, выборку его содержимого и активизацию ожидающего его процесса уходят десятки миллисекунд. Задержки такого порядка отодвигают на второй план все остальные факторы ухудшения производительности. Каждое обращение к диску вызывает приостановку активного процесса "стоимостью" несколько сотен миллионов инструкций ЦП.

Поскольку в Linux используется виртуальная память, скорость дискового обмена непосредственно связана с объемом имеющейся памяти. В сильно загруженной системе с ограниченным объемом ОЗУ для получения чистой страницы виртуальной памяти часто приходится записывать ее прежнее содержимое на диск. К сожалению, это означает, что обращение к памяти по своей "стоимости" приближается к работе с диском. Страничный обмен, вызванный непомерными объемами выполняющихся программ, для большинства рабочих станций является врагом производительности номер один.

Пропускная способность сети подвержена примерно тем же ограничениям, что и скорость дискового обмена. Ее ухудшение связано со всевозможными задержками. Но возникающие проблемы охватывают целые сообщества пользователей, а не отдельные компьютеры. Кроме того, сети восприимчивы к проблемам аппаратного обеспечения и перегрузкам серверов.

25.3. Проверка производительности системы

Программы анализа производительности в большинстве своем сообщают о том, что происходит в системе в данный момент времени. Но следует учесть, что структура и характер нагрузки меняются в течение дня. Поэтому до принятия мер обязательно проведите всесторонний анализ данных, касающихся производительности. Истинная производительность выясняется только после длительного (месяц или даже больше) наблюдения за системой. Особенно важно собирать данные в периоды пиковой загруженности системы. Ограничения на использование ресурсов и ошибки в конфигурации системы часто выявляются только в таких условиях.

Анализ использования центрального процессора

Обычно собирают три вида данных о центральном процессоре: общий показатель использования, показатели средней загруженности и потребление ресурсов отдельными процессами. Первая характеристика определяет, является ли быстродействие самого процессора узким местом. Показатели средней загруженности дают возможность оценить общую производительность системы. Данные, касающиеся отдельных процессов, позволяют выявить наиболее ресурсоемкие процессы.

Сводную информацию выдает команда `vmstat`. Она принимает два аргумента: время (в секундах), в течение которого необходимо наблюдать за системой для получения каждой строки выходной информации, и необходимое количество отчетов. Если не указать число отчетов, команда будет выполняться до тех пор, пока пользователь не нажмет <Ctrl+C>. В первой строке отображаемых данных сообщаются средние значения, измеренные с момента запуска системы. В последующих строках выдаются результаты по каждому очередному замеру, стандартная продолжительность которого составляет 5 секунд. Например:

```
$ vmstat 5 5
procs  ----------memory-------- -swap-----io---- --system--  ----cpu----
r b swpd     free  buff     cache si so    bi  bo    in    cs  us  sy  id  wa
1 0  820  2606356  428776  487092  0  0  4741  65  1063  4857  25   1  73   0
1 0  820  2570324  428812  510196  0  0  4613  11  1054  4732  25   1  74   0
1 0  820  2539028  428852  535636  0  0  5099  13  1057  5219  90   1   9   0
1 0  820  2472340  428920  581588  0  0  4536  10  1056  4686  87   3  10   0
3 0  820  2440276  428960  605728  0  0  4818  21  1060  4943  20   3  77   0
```

Пользовательское время, системное время (время ядра), время простоя и время ожидания для каналов ввода-вывода (I/O) показаны в колонках us, sy, id и wa, соответственно. Большие числа в колонке us обычно означают вычисления, а в колонке sy они свидетельствуют о том, что процессы осуществляют очень много системных вызовов или выполняют операции ввода-вывода.

Выработанное нами за многие годы эмпирическое правило для серверов общего назначения, справедливое для большинства систем, гласит: система должна тратить примерно 50% рабочего времени на обслуживание пользовательских запросов и столько же — на системные запросы. Общее время простоя не должно быть нулевым. Если сервер выделяется под одно единственное, но интенсивно использующее ЦП приложение, большая часть времени должна тратиться на обработку пользовательских запросов.

В колонке cs показано число переключений контекста за данный период, т.е. сколько раз ядро переключало процессы. В колонке in отображается число прерываний, генерируемых аппаратными устройствами или компонентами ядра. Слишком большие значения в этих колонках свидетельствуют о неправильно работающем аппаратном устройстве. Остальные колонки важны для анализа использования памяти и жесткого диска, о чем будет рассказываться позже в этой главе.

Длительные измерения статистики работы центрального процессора позволяют определить, достаточно ли его ресурсов для нормальной работы системы. Если процессор регулярно часть времени простаивает, значит, есть запас по тактовой частоте. В этом случае увеличение быстродействия процессора ненамного улучшит общую производительность системы, хотя может и ускорить выполнение отдельных операций.

Как видно из показанного примера, центральный процессор постоянно переключается из режима интенсивного использования в режим полного бездействия и обратно. Поэтому необходимо наблюдать за показателями, соответствующими обоим режимам, и выводить среднее за определенный период времени. Чем меньше интервал наблюдения, тем ниже достоверность оценок.

В мультипроцессорных системах команды вроде vmstat сообщают средние значения по всем процессорам. Но существует и команда mpstat, которая выдает отчет по каждому процессору. С помощью флага -P можно выбрать один конкретный процессор. Этой командой удобно пользоваться при отладке программ, поддерживающих симметричную многопроцессорную обработку. Интересно также узнать, насколько система эффективно (или неэффективно) работает с несколькими процессорами.

Центральный процессор однопользовательской рабочей станции обычно простаивает 99% времени. Когда запрашивается прокрутка содержимого окна, ЦП справляется с этой операцией за считанные секунды. В такой ситуации долгосрочный средний показатель использования ЦП практически лишен смысла.

Вторым статистическим показателем, полезным для оценки интенсивности использования системы является показатель “средняя загруженность”, который представляет собой среднее количество выполняемых процессов. Он дает достаточное представление о том, сколько процессов требуют свою долю процессорного времени. Узнать его значение можно с помощью команды uptime:

```
$ uptime
11:10am  up 34 days, 18:42, 5 users, load average: 0.95, 0.38, 0.31
```

Выдаются три значения, которые соответствуют средней загруженности за пять, десять и пятнадцать минут. Как правило, чем выше средняя загруженность, тем важнее общая производительность системы. Если выполняется всего лишь один процесс, то он

обычно ограничен одним ресурсом (центральным процессором или пропускной способностью дискового канала ввода-вывода). Пиковый спрос на этот ресурс и становится определяющим фактором производительности.

Когда в системе одновременно работает несколько процессов, нагрузка распределяется более равномерно. Если все работающие процессы потребляют ресурсы ЦП, диска и памяти, то зависимость производительности системы от ограниченных возможностей какого-то одного ресурса снижается. В такой ситуации гораздо важнее следить за средними показателями загруженности, в частности за общим временем использования центрального процессора.

Обычно в Linux-системах значение 3 у показателя средней загруженности свидетельствует о сильной загруженности системы, а значение 8 и выше — вообще уже считается критическим. Такие значения являются сигналом о том, что пора начать поиск путей искусственного перераспределения нагрузки, например назначить процессам приоритеты с помощью команды `nice`.

📖 О приоритетах рассказывалось в конце раздела 4.1.

Показатель средней загруженности — отличная метрика для повседневного контроля системы. Если известен показатель нормально работающей системы и при возникновении проблем он находится в том же диапазоне, значит, следует искать внешний источник проблемы (это может быть, к примеру, сеть). В противном случае нужно проверить процессы самой Linux-системы.

Еще один способ контроля над использованием ЦП — посмотреть, какую часть его времени отнимает каждый процесс. Для этого служит команда `ps -aux`. Чаще всего оказывается, что в интенсивно эксплуатируемой системе минимум 70% времени ЦП отнимается одним-двумя процессами (помните о том, что команда `ps` сама занимает немного ресурсов). Запуск таких процессов в другое время суток или снижение их приоритетов позволит высвободить ЦП для выполнения других процессов.

Прекрасной альтернативой команде `ps` является утилита `top`. Она выдает примерно ту же информацию, что и `ps`, но в "живом" формате, позволяющем наблюдать за изменением состояния системы во времени[2].

📖 Подробнее об утилите `top` рассказывалось в разделе 4.8.

Управление памятью в Linux

Прежде чем поговорить об особенностях системы управления памятью в Linux, не помешает еще раз напомнить о том, что ядро Linux совершенствуется гораздо быстрее, чем практически любая другая операционная система. За последние годы подсистема управления памятью в Linux изменялась особенно часто (даже по меркам Linux).

Ядро версии 2.6 включает значительно обновленную систему VM[3]. Однако на том уровне, на котором ведется обсуждение в этой главе, точная версия ядра особого значения не имеет. Описываемые здесь концепции не зависят от версии ядра.

---

[2] Утилита `top` потребляет довольно много ресурсов, поэтому пользуйтесь ею в разумных пределах.

[3] Ядро версии 2.6 включает опцию, позволяющую запускать систему без виртуальной памяти. Эта опция особенно полезна для встраиваемых систем. Решив запустить систему без виртуальной памяти, следует убедиться в том, что физической памяти хватит для удовлетворения всех потребностей, которые могут возникнуть.

Как и в других UNIX-системах, в Linux память организована в виде модулей, называемых *страницами*. На персональных компьютерах размер страницы памяти в настоящий момент составляет 4 Кбайт[4]. Когда процессы запрашивают у операционной системы память, ядро Linux выделяет им виртуальные страницы. Каждая такая страница отображается на настоящее физическое хранилище, такое как ОЗУ или "резервное ЗУ" на жестком диске. (Резервное ЗУ обычно представляет собой просто пространство в области подкачки, но для страниц, которые содержат текст исполняемой программы, резервное ЗУ — это самый настоящий исполняемый файл.) Информация о связях между виртуальными и реальными страницами хранится в *таблице страниц*.

Ядро Linux способно эффективно обслуживать запросы процессов, дополняя реальное ОЗУ областью подкачки. Системе постоянно приходится перемещать страницы между ОЗУ и этой областью. Такие операции называются *страничным обменом* (paging) [5].

Linux старается управлять памятью так, чтобы страницы, к которым недавно обращались, хранились в физической памяти, а менее активные страницы выгружались на диск. Этот алгоритм известен под названием LRU (least recently used — замещение наименее используемых элементов), поскольку те страницы, к которым долго никто не обращался, перемещаются на диск. Впрочем, отслеживать все обращения к страницам было бы слишком накладно для ядра, поэтому в Linux используется страничный кэш-буфер, на основании анализа содержимого которого и принимаются решения о выгрузке страниц. Такая система гораздо проще в реализации, а результаты дает почти такие же.

Linux отслеживает "возраст" каждой страницы виртуальной памяти. Всякий раз, когда происходит обращение к странице, ее "возраст" увеличивается. (В этом смысле термин "возраст" не совсем удачен, поскольку значение параметра отражает частоту доступа. Чем выше "возраст", тем новее страница.) Параллельно с этим выполняется демон `kswapd`, который регулярно уменьшает "возраст" неактивных страниц.

Ядро ведет несколько списков страниц. Страницы, "возраст" которых больше нуля, считаются активными и помещаются в список активных страниц (CAC). С другой стороны, как только "возраст" снижается до нуля, демон `kswapd` перемещает страницу в список неактивных страниц (CHC) и удаляет ссылку на нее из таблицы страниц. Такие страницы считаются готовыми к выгрузке на диск, однако при поступлении запроса к ним ядро сможет восстановить их из памяти или с диска и снова поместить в таблицу страниц.

В случае нехватки памяти ядро пытается определить, какие страницы из CHC использовались наиболее давно. Если при последнем обращении такие страницы модифицировались процессом, ядро считает их "грязными"; они обязательно должны быть выгружены на диск, прежде чем память будет повторно использована. Все остальные страницы считаются "чистыми" и могут быть переданы другому процессу.

Когда выполняется обращение к странице из CHC, ядро возвращает ссылку на нее в таблицу страниц, обнуляет ее "возраст" и переводит страницу из CHC в CAC. Страницы, выгруженные на диск, должны быть прочитаны с диска, прежде чем их можно будет активизировать. *Ошибка программной страницы* происходит, когда процесс обращается к странице CHC, находящейся в памяти. *Ошибка страницы диска* имеет место в случае не-

[4] В ядре версии 2.6 также можно использовать страницы размером 4 Мбайт, установив переменную `HPAGE_SIZE`.

[5] Обратите внимание на то, что "страничный обмен" и "подкачка" реализуются одинаковым образом и подразумевают использование одной и той же области "подкачки" для сохранения страниц, когда страницы находятся не в ОЗУ.

резидентной (выгруженной) страницы. Другими словами, ошибка второго типа требует прочитать страницу с диска.

Потребность в памяти меняется, поэтому ядро может запускать алгоритм определения возраста страниц `kswapd` с разной частотой. Когда спрос на память крайне высок, алгоритм запускается часто, поэтому страницам удается оставаться в памяти только при очень частом обращении.

В вопросе сортировки страниц на активные и неактивные подсистема виртуальной памяти зависит от промежутка времени, которое проходит с момента помещения страницы в СНС до момента выгрузки ее на диск. Следовательно, для выбора соответствующей скорости вызова алгоритма `kswapd` системе приходится прогнозировать будущую активность страничного обмена. Если этот алгоритм будет вызываться реже, чем нужно, то "чистых" неактивных страниц может не хватить для обеспечения достаточного объема памяти. Если же он, наоборот, будет вызываться слишком часто, ядру придется тратить лишнее время на обработку ошибок программных страниц.

Поскольку алгоритм страничного обмена является прогнозирующим, операции выделения и выгрузки страниц не обязательно взаимосвязаны. Цель системы — обеспечить наличие достаточного объема свободной памяти, чтобы процессам не приходилось ждать выгрузки страниц всякий раз, когда им нужна свободная память. Если в периоды сильной загруженности системы страничный обмен резко возрастает, имеет смысл купить дополнительную память.

Можно настроить параметр "подкачки" (`/proc/sys/vm/swappiness`) и тем самым дать ядру подсказку о том, насколько быстро оно должно делать страницы пригодными для возврата из процесса в случае нехватки памяти. По умолчанию для этого параметра устанавливается значение 60. Если для него установить значение 0, ядро будет забирать страницы, которые были назначены процессу, только тогда, когда испробует все остальные возможности. Если для него установить значение 60 или выше (максимальным значением является 100), ядро, скорее всего, будет забирать страницы из процесса только тогда, когда исчерпаются все запасы памяти. В любом случае, запомните, что если возникает желание изменить этот параметр, значит, возможно, пришла пора увеличить ОЗУ.

Когда ядру не хватает как физической памяти, так и области подкачки, это означает, что виртуальная память исчерпана. В данной ситуации включается режим "принудительного уничтожения". Чтобы освободить память, системе приходится уничтожать целые процессы. И хотя выбираются наименее важные для системы процессы, все равно это очень неприятная процедура, ведь значительная часть ресурсов системы тратится не на полезную работу, а на управление памятью.

Проблемы со страницами могут возникнуть даже у процессов, работающих с низким приоритетом. Предположим, на рабочей станции выполняется клиент SETI[6] с очень низким приоритетом (и, соответственно, высоким значением `nice`). Одновременно с этим пользователь читает электронную почту. Как только пользователь делает паузу, чтобы прочесть сообщение, показатель использования центрального процессора падает до нуля и клиент получает разрешение на работу. Он активизирует все свои страницы, вытесняя из памяти страницы интерпретатора команд, диспетчера окон, программы чтения почты и даже эмулятора терминала. Когда пользователь нажимает клавишу или щелкает мышью для перехода к следующему сообщению, возникает длинная пауза, по-

[6] Информацию о проекте SETI (Search for Extraterrestrial Intelligence — поиск внеземных цивилизаций) можно найти по адресу `www.seti.org/science/setiathome.html`.

тому что системе приходится подгружать большой блок памяти. То есть на практике высокое значение nice еще не гарантирует, что процесс с низким приоритетом не вызовет ухудшения производительности.

Анализ использования памяти

Интенсивность операций с памятью количественно представляется тремя показателями: общим объемом задействованной виртуальной памяти, скоростью подкачки и выгрузки страниц. Первый показатель характеризует общую потребность в памяти, а следующие два указывают на то, какая доля этой памяти активно используется. Задача администратора заключается в снижении интенсивности использования или увеличении объема памяти до тех пор, пока не будет достигнут приемлемый уровень страничного обмена. Страничный обмен — процесс неизбежный, поэтому не пытайтесь полностью избавиться от него.

Определить объем доступной памяти и размер области подкачки позволяет команда free. При наличии флага -t будет автоматически вычисляться суммарный объем виртуальной памяти.

```
$ free -t
             total       used       free     shared    buffers   cached
Mem:        127884      96888      30996      46840      57860    10352
-/+ buffers/cache:      28676      99208
Swap:       265032       3576     261456
Total:      392916     100464     292452
```

В колонке free указан объем (в килобайтах) списка неиспользуемых страниц. Если приведенные в ней показатели не достигают 3% от общего объема системной памяти, это свидетельствует о наличии проблем. С помощью команды swapon можно узнать, какие именно файлы и разделы задействованы под область подкачки:

```
$ swapon -s
Filename                Type            Size    Used    Priority
/dev/sdb7               partition       265032  3576    -1
```

В данной системе это единственный дисковый раздел: sdb7. Поскольку страничный обмен может оказывать существенное влияние на скорость работы с диском, почти всегда лучше распределять область подкачки между несколькими физическими дисками.

В традиционных UNIX-системах есть команда vmstat, которая выдает информацию о страничном обмене и областях подкачки. Однако гораздо лучшим источником информации является команда procinfo, входящая в большинство дистрибутивов Linux. Она не всегда инсталлирована по умолчанию, поэтому, возможно, придется вручную установить соответствующий пакет.

📖 О файловой системе /proc более подробно будет рассказываться в разделе 28.4.

Команда procinfo самостоятельно не осуществляет никаких замеров и вычислений. Она просто форматирует данные, извлекаемые из элементов файловой системы /proc. Без ее помощи интерпретировать соответствующие файлы было бы нелегко. Если нужно получать отчет, обновляющийся, скажем, каждые 5 секунд, введите команду procinfo -n5.

```
$ procinfo
Linux 2.6.9-34.EL (root@bull) (gcc 3.4.3 20050227 ) #1 1CPU [main]
```

```
Memory:      Total       Used       Free     Shared    Buffers
Mem:        463728     371512      92216          0     144042
Swap:      1866470        218    1626252

Bootup: Sat Jul 29 08:47:20 2006 Load average: 0.08 0.06 0.00

user   :       1:20:32.12   1.3%     page in :   0
nice   :       0:02:36.30   0.0%     page out :   0
system :       3:06:46.90   0.8%      swap in :   0
idle   :    1d 7:23:49.50  97.9%     swap out :   0
steal  :       0:00:00.00   0.0%
uptime :    4d 1:51:43.64             context :   55465717

irq 0:  5748875398  timer         irq  8:          3  rtc
irq 1:           3                irq 11:   75300822  eth0
irq 2:           0  cascade[4]    irq 12:          3
irq 4:           4                irq 14:    7529735  ide0
```

Часть выдаваемой информации можно получить с помощью команд `free`, `uptime` и `vmstat`. Команда `procinfo` сообщает также сведения о версии ядра, страничном обмене, доступе к диску и векторах прерываний. Команда `procinfo -a` извлекает дополнительные данные из файловой системы `/proc`, включая параметры загрузки ядра, списки загружаемых модулей ядра, устройств посимвольного ввода-вывода и файловых систем.

Кажущиеся несоответствия между колонками большей частью иллюзорны. В одних колонках указывается число страниц, в других — объем в килобайтах. Все значения являются округленными средними величинами. Более того, некоторые из них — средние скалярных величин, а некоторые — средние изменений. Например, нельзя вычислить следующее значение `Free` по текущему значению этого параметра и имеющимся сведениям о страничном обмене, потому что события, которые определяют следующее среднее значение `Free`, еще не произошли.

С помощью показателей `page in/out` и `swap in/out` можно оценить интенсивность подкачки и страничного обмена. Операция загрузки (`page in`) не обязательно означает, что из области подкачки восстанавливается страница. Это может быть исполняемый код, постранично загружаемый из файловой системы, или копия страницы, создаваемая в режиме дублирования при записи. Оба этих случая вполне типичны и не обязательно означают нехватку памяти. С другой стороны, операция выгрузки (`page out`) всегда означает принудительное изъятие данных ядром.

Система, в которой непрерывно происходят операции выгрузки, скорее всего, выиграет от добавления памяти. Если же это случается лишь время от времени и не вызывает жалоб со стороны пользователей, то можно не беспокоиться. В остальных случаях дальнейший анализ будет зависеть от того, что нужно сделать: оптимизировать систему для интерактивного режима (например, как рабочую станцию) или сконфигурировать ее для одновременной работы пользователей (например, как сервер приложений).

Если половину операций страничного обмена составляет выгрузка, можно считать, что каждые 100 таких операций создают задержку примерно в одну секунду. Соответственно, если для прокрутки окна требуется 150 операций выгрузки, придется ждать около полутора секунд. Исследователи пользовательских интерфейсов утверждают, что среднестатистический пользователь считает систему “медленной”, когда время ее реакции превышает семь десятых секунды.

Следует также отметить, что команда `procinfo` выдает информацию о центральном процессоре, недоступную командам `vmstat` и `uptime`. Она определяет показатель средней загруженности за последние 5, 10 и 15 минут, а также сообщает количество выпол-

няемых в данный момент процессов, общее число процессов и, наконец, идентификатор самого последнего процесса. Приведем фрагмент результатов, сообщаемых командой `procinfo` на загруженном сервере:

```
Load average:  2.37  0.71  0.29  3/67  26941
```

Общее число процессов на данном сервере — 67, из которых запускалось только три. Последний запущенный процесс имел идентификатор 26941 (в данном случае это процесс интерпретатора команд, из которого была вызвана команда `procinfo`).

Анализ операций обмена с диском

Производительность жесткого диска можно контролировать с помощью команды `iostat`. Как и `vmstat`, эта команда поддерживает дополнительные аргументы, задающие интервал времени в секундах между моментами выдачи статистических данных за истекший период и число повторений. Команда `iostat` выдает также сведения об использовании ЦП. Приведем небольшой пример, в котором показана только информация, касающаяся жестких дисков:

```
$ iostat
...
Device:     tps   Blk_read/s  Blk_wrtn/s   Blk_read   Blk_wrtn
hdisk0     0.54         0.59        2.39     304483    1228123
hdisk1     0.34         0.27        0.42     140912     216218
hdisk2     0.01         0.02        0.05       5794      15320
hdisk3     0.00         0.00        0.00          0          0
```

Команда `iostat` собирает данные из файловой системы `/proc`, выдавая по одной строке для каждого физического диска системы. К сожалению, Linux хранит лишь минимальную статистическую информацию об использовании жестких дисков, и даже к этой информации доступ ограничен. Для каждого жесткого диска сообщается число операций ввода-вывода в секунду (`tps`), количество операций блочного чтения и блочной записи в секунду (`Blk_read/s` и `Blk_write/s`), общий объем прочитанных (`Blk_read`) и записанных (`Blk_wrtn`) блоков. Если в системе меньше четырех дисков, отображаются только строки-заполнители, состоящие из одних нулей.

Размер дискового блока обычно равен 1 Кбайт, поэтому можно легко определить реальную скорость обмена данными с диском в килобайтах в секунду. В то же время понятие операции ввода-вывода несколько размыто. Один запрос на пересылку данных может включать в себя несколько запросов ввода-вывода к различным секторам.

Время, затрачиваемое на поиск блока, — основной фактор, влияющий на производительность жесткого диска. В первом приближении скорость вращения диска и быстродействие шины, к которой он подключен, не имеют особого значения. Современные диски могут пересылать десятки мегабайт данных в секунду, если эти данные считываются из смежных секторов. Вместе с тем количество операций поиска составляет от 100 до 300 в секунду. Если после каждой операции поиска будет читаться один-единственный сектор, легко подсчитать, что в таком режиме задействуется менее 5% максимальной пропускной способности канала обмена данными с диском.

Затраты времени на поиск блока возрастают, если головке приходится перемещаться на большие расстояния. Когда есть диск с файловой системой, расположенной в нескольких разделах, и файлы считываются из каждого раздела в случайной последовательности, то для перехода из одного раздела в другой головка вынуждена преодолевать очень большой путь. С другой стороны, файлы в одном разделе расположены относи-

тельно близко друг к другу. Разбивая новый диск на разделы, необходимо принять во внимание факторы, влияющие на производительность, и постараться поместить файлы, обращение к которым осуществляется одновременно, в одну файловую систему.

Для достижения максимальной производительности нужно помещать файловые системы, которые используются одновременно, на разные диски. Хотя тип шины и драйверы устройств влияют на степень эффективности, большинство компьютеров может работать с несколькими дисками параллельно, что значительно повышает пропускную способность дисковой подсистемы. Например, данные и журнальные файлы Web-сервера имеет смысл размещать на разных дисках.

Особенно важно распределить область подкачки между несколькими дисками, если это возможно, поскольку страничный обмен обычно замедляет работу системы в целом. Такую конфигурацию можно реализовать с помощью команды `swapon`. Ядро Linux позволяет создавать как выделенные разделы подкачки, так и файлы подкачки, записываемые в обычную файловую систему.

Linux-системы позволяют создавать "резидентные" файловые системы. Фактически это то же самое, что и виртуальный диск ПК. Специальный драйвер выдает себя за драйвер диска, хотя на самом деле записывает данные в память. Во многих организациях виртуальные диски применяются для хранения файловой системы `/tmp` и других часто используемых файлов, например журналов Web-сервера и почтовых буферов. Это приводит к уменьшению объема общедоступной памяти, но значительно ускоряет чтение и запись временных файлов. Как правило, такой подход оказывается весьма эффективным.

Большинство ядер Linux компилируется с поддержкой виртуальных дисков. В каталоге `/dev` содержатся файлы устройств для таких дисков, например `/dev/ram0` и `/dev/ram1`. Количество поддерживаемых устройств неодинаково, но обычно их не менее пяти.

Для подключения виртуального диска необходимо сначала отформатировать файловую систему на свободном виртуальном дисковом устройстве, а затем смонтировать эту файловую систему в существующем каталоге:

```
# mke2fs /dev/ram12
# mount /dev/ram12 /tmp/fastdisk
```

По умолчанию размер виртуального диска составляет всего 4 Мбайт, что недостаточно для каталога `/tmp`. К сожалению, процедура изменения размера не так проста, поскольку размер задается в виде переменной ядра. Можно либо добавить строку вида `ramdisk_size=100000` (указывается количество блоков размером 1 Кбайт) в файл конфигурации системного загрузчика, чтобы новое значение устанавливалось на этапе начальной загрузки, либо сделать драйвер виртуального диска динамически загружаемым модулем. В последнем случае нужно добавить необходимый аргумент в файл `/etc/modules.conf` или воспользоваться командой `insmod`.

Некоторые системные компоненты замедляют выполнение основных операций и тем самым ухудшают производительность системы. В качестве двух примеров приведем дисковые квоты и подсистему учета использования центрального процессора. Квоты требуют изменения статистической сводки использования блоков по мере записи и удаления файлов, а подсистема учета предполагает запись данных в учетный файл после завершения каждого процесса. Кэширование диска помогает смягчить влияние этих факторов, однако оно все равно может быть весьма ощутимым, поэтому ни квоты, ни подсистема учета обычно не используются.

Выбор планировщика ввода-вывода

Алгоритм планирования подсистемы ввода-вывода выступает в роли "арбитра" между соревнующимися за дисковый ввод-вывод процессами. Он оптимизирует порядок и время запросов для обеспечения наилучшей возможной производительности подсистемы ввода-вывода в целом.

В ядро 2.6 встроены четыре разных алгоритма планирования. Можно выбирать любой. К сожалению, алгоритм планирования устанавливается при помощи атрибута ядра `elevator=`*`алгоритм`* и поэтому может быть установлен только во время загрузки. А потому он обычно указывается в файле `grub.config`.

Все доступные алгоритмы перечислены ниже.

- Comletely Fair Queuing (`elevator=cfq`). Этот алгоритм используется по умолчанию и обычно является наиболее подходящим вариантом для серверов общего назначения. Он старается равномерно предоставлять доступ к подсистеме ввода-вывода.
- Deadline (`elevator=deadline`). Этот алгоритм старается сводить к минимуму время задержки для каждого запроса. Он изменяет порядок запросов для повышения производительности.
- NOOP (`elevator=noop`). Этот алгоритм реализует простую очередь типа FIFO (First In, First Out — первым пришел, первым обслужен). Он считает, что запросы к подсистеме ввода-вывода оптимизирует или переупорядочивает драйвер или устройство (каковым вполне может быть какой-нибудь контроллер со встроенной логикой). Он может быть наиболее подходящим вариантом в некоторых SAN-средах.
- Anticipatory (`elevator=as`). Этот алгоритм откладывает запросы к подсистеме ввода-вывода в надежде, что они будут упорядочены наиболее эффективным образом. Он может оказаться неплохим вариантом для настольной рабочей станции, но редко бывает оптимальным выбором для сервера.

Определение того, какой из этих алгоритмов планирования является наиболее подходящим для данной среды (которое может потребовать перепробовать все четыре варианта), возможно, позволит напрямую повлиять на производительность подсистемы ввода-вывода.

`sar`: сбор статических данных и генерирование отчетов по ним

Одним из наиболее часто используемых сегодня на системах Linux средств для мониторинга производительности является команда `sar`. Эта команда появилась еще во времена AT&T UNIX, но с тех пор была значительно переделана для использования под Linux.

На первый взгляд может показаться, что `sar` отображает ту же информацию, что и `procinfo`, `vmstat` и `iostat`. Однако имеется одно очень важное отличие: `sar` умеет предоставлять отчеты как по старым (накопленным), так и по текущим данным.

Без опций команда `sar` предоставляет отчет о том, как ЦП использовался в течение того или иного дня каждые 10 минут, начиная с полуночи. Получение такой коллекции накопленных данных делает возможным сценарий `sal`, который является частью пакета `sar` (иногда также называемого пакетом `sysstat`) и требует указания интервала време-

ни, через который он должен запускаться из cron. Все данные, которые sar собирает, она сохраняет в /var/log/sa в бинарном формате.

```
$ sar
Linux 2.6.9-11.ELsmp (bull.atrust.com) 08/04/2006

12:00:01 AM       CPU     %user     %nice   %system   %iowait    %idle
12:10:01 AM       all      0.10      0.00      0.04      0.06    99.81
12:20:01 AM       all      0.04      0.00      0.03      0.05    99.88
12:30:01 AM       all      0.04      0.00      0.03      0.04    99.89
12:40:01 AM       all      0.09      0.00      0.03      0.05    99.83
12:50:01 AM       all      0.04      0.00      0.03      0.04    99.88
01:00:01 AM       all      0.05      0.00      0.03      0.04    99.88
```

Помимо информации о ЦП sar также может предоставлять отчеты по метрическим показателям, таким как показатели дисковой и сетевой активности. sar -d отображает сводку данных об активности диска за сегодняшний день, sar -n DEV — статистические данные о сетевом интерфейсе, а sar -A — всю доступную информацию.

sar имеет некоторые ограничения и потому хорошо подходит только для быстрого получения кратких накопленных сведений. Тем, кто решил "всерьез и надолго" заняться мониторингом производительности, лучше установить специальную платформу с возможностями сбора коллекций данных и представления их в виде графиков, такую как платформа Cacti. Cacti пришла к нам из мира сетевого управления, но действительно умеет отображать произвольные метрические показатели системы, такие как показатели использования ЦП и памяти, в виде графиков.

📖 Пример отображаемого Cacti графика приводился в разделе 19.10.

oprofile: универсальный профилировщик

oprofile — это чрезвычайно мощная, интегрируемая в систему программа профилирования для систем Linux с ядром версии 2.6 или выше. Профилированию могут подвергаться все компоненты системы Linux: аппаратные и программные обработчики прерываний, модули ядра, само ядро, совместно используемые библиотеки и приложения.

При наличии массы свободного времени и желания точно знать, как используются ресурсы системы (до самых мельчайших деталей), стоит рассмотреть вариант установки oprofile. Эта утилита особенно полезна для тех, кто у себя дома разрабатывает свои собственные приложения или код ядра.

В состав дистрибутива oprofile, который доступен для загрузки на сайте sourceforge.net, входят как модули ядра, так и ряд пользовательских утилит.

В Red Hat для получения файлов, необходимых для запуска oprofile, придется установить пакет kernel-debuginfo.

25.4. Помогите! Моя система почти остановилась!

В предыдущих разделах речь шла в основном о средней производительности системы. Для ее повышения требуется корректировать конфигурационные параметры или модернизировать систему.

Но даже правильно сконфигурированные системы иногда работают медленнее обычного. К счастью, нерегулярные проблемы обычно легко диагностируются. В 90% случаев они создаются "ненасытным" процессом, который потребляет так много ресурсов про-

цессора, жесткого диска или сетевой подсистемы, что остальные процессы буквально останавливаются. Иногда процесс намеренно захватывает ресурсы, чтобы замедлить работу системы или сети. Это называется атакой типа "отказ в обслуживании".

Очень часто для определения того, какой конкретно ресурс исчерпан, не нужно даже запускать диагностическую программу. Если система начинает "подвисать"[7] или подозрительно долго обращаться к диску, то проблема, скорее всего, связана с пропускной способностью дисковой подсистемы или дефицитом памяти. Если же производительность приложений падает ниже критического уровня, проблема, возможно, заключается в загрузке ЦП.

Первый шаг в диагностировании проблемы — запуск команды `ps auxww` или `top` для выявления явно неуправляемых процессов. Любой процесс, отнимающий более 50% времени ЦП, можно с большой долей вероятности считать ненормальным. Если столь непомерную долю ресурсов ЦП не получает ни один процесс, посмотрите, сколько процессов получают минимум по 10%. Когда их больше двух-трех (не считая самой команды `ps`), средняя загруженность скорее всего будет очень высокой. Такая ситуация сама по себе является причиной низкой производительности. Проверьте среднюю загруженность системы с помощью команды `uptime`, а затем выполните команду `vmstat` или `top`, чтобы узнать, простаивает ли когда-нибудь процессор.

Если конкуренции за право использования центрального процессора не наблюдается, выполните команду `vmstat` или `procinfo`, чтобы посмотреть, какова интенсивность страничного обмена. Следует обращать внимание на все операции обмена с диском: множество операций выгрузки может означать соперничество за память, а наличие дискового трафика в отсутствие страничного обмена говорит о том, что какой-то процесс монополизировал диск, непрерывно читая и записывая файлы.

Нельзя непосредственно сопоставить дисковые операции с выполняющими их процессами, но с помощью команды `ps` можно сузить круг подозреваемых. Любой процесс, работающий с диском, должен отнимать какую-то часть времени ЦП. Как правило, всегда можно сделать обоснованное предположение о том, какой конкретно из активных процессов захватил ресурсы[8]. Для проверки своей теории на практике выполните команду `kill -STOP`.

Предположим, процесс-виновник найден. Что с ним делать дальше? Как правило, ничего. Некоторые операции действительно требуют много ресурсов и неизбежно замедляют работу системы, но это вовсе не означает, что они незаконны. С помощью команды `renice` можно изменить приоритет процесса, для которого ограничивающим фактором является быстродействие ЦП, но не забудьте попросить владельца этого процесса впоследствии выполнить команду `nice`. Иногда правильная настройка приложения может привести к значительному снижению потребления ресурсов. Этот эффект особенно заметен в сетевых серверных программах, например в Web-приложениях.

Процессы, интенсивно использующие диск и память, нелегко поддаются воздействию. Команда `renice` обычно не помогает. Можно уничтожить или остановить процесс,

[7] То есть требуется больше времени на переключение между приложениями, хотя скорость выполнения отдельных заданий приемлема.

[8] Раньше подозрительными признаками служили большой размер области данных процесса, размещенной в виртуальной памяти, либо неестественно большой объем занимаемой процессом физической памяти, но появление совместно используемых библиотек сделало эти показатели не столь полезными. Команда `ps` не умеет отделять общесистемные совместно используемые библиотеки от адресного пространства отдельных процессов. Кажется, что многие процессы занимают десятки мегабайт физической памяти, хотя на самом деле это не так.

но, на наш взгляд, такая мера оправдана только в экстренных ситуациях. Лучше прибегнуть к административной мере: попросить владельца запустить процесс позже.

Ядро позволяет процессу ограничивать объем потребляемой им самим физической памяти при помощи системного вызова `setrlimit`. Эта возможность также доступна в оболочке C в виде встроенной команды `limit`. Например, команда

```
% limit memoryuse 32m
```

ограничивает использование физической памяти для всех последующих пользовательских команд тридцатью двумя мегабайтами. Ее действие примерно эквивалентно действию команды `renice` в отношении процессов, ограничивающим фактором для которых является объем памяти. Можно тактично предложить пользователям вставить такую строку в свои файлы `.cshrc`.

Если неуправляемый процесс не является источником снижения производительности, нужно проанализировать две другие возможные причины. Первая — перегрузка сети. Многие программы настолько тесно связаны с сетью, что иногда трудно понять, где речь идет о производительности системы, а где — о производительности сети. Подробная информация о средствах контроля над работой сети приведена в главе 19.

В некоторых случаях проблемы, связанные с перегрузкой сети, выявить сложно, потому что они возникают и исчезают очень быстро. Например, если на всех компьютерах сети каждый день в определенное время с помощью демона `cron` запускается какая-то сетевая программа, то в сети может возникать кратковременный, но серьезный затор. Все компьютеры "зависнут" секунд на пять, после чего ситуация нормализуется.

Еще одной причиной кризисов производительности являются задержки, связанные с серверами. Linux-системы постоянно обращаются к удаленным серверам NFS, NIS, DNS и т.п. Если сервер не работает или какая-то иная проблема привела к тому, что связь с ним стала ненадежной, это ощущают на себе все клиентские системы.

Предположим, что в интенсивно эксплуатируемой системе какой-то процесс каждые несколько секунд вызывает библиотечную функцию `gethostent`. Если сбой в работе DNS заставляет эту функцию тратить на свое выполнение две секунды, будет заметно общее снижение производительности системы. На удивление большое число проблем с производительностью серверов связано с неправильной конфигурацией прямого и обратного поиска в DNS.

25.5. Рекомендуемая литература

- Ezolt, Phillip G. *Optimizing Linux Performance*. Upper Saddle River, NJ: Prentice Hall PTR, 2005.
- Johnson, S., et al. *Performance Tuning for Linux Servers*. Indianapolis, IN: IBM Press, 2005.
- Loukides, Mike and Gian-Paolo D. Musumeci. *System Performance Tuning .2nd Edition*. Sebastopol: O'Reilly & Associates, 2002.

25.6. Упражнения

25.1. Определите возможную причину проблем в каждом из нижеперечисленных случаев.

а) При переключении между приложениями возникает заметная пауза и слышно, как диск интенсивно перекачивает данные.

б) Программа численного моделирования выполняется дольше, чем нужно, но системная память практически свободна.

в) Пользователи интенсивно эксплуатируемой ЛВС жалуются на медленный доступ через NFS, но показатель средней загруженности сервера остается очень низким.

г) В процессе работы команда часто сообщает о нехватке памяти.

25.2. Выполните команду `procinfo` на компьютере, работающем под управлением Linux, и проанализируйте результаты. Какие ресурсы используются интенсивнее всего? Какие ресурсы вообще не задействованы?

☆ 25.3. Системы балансирования нагрузки способны существенно влиять на производительность серверов. Перечислите несколько механизмов, которые могут использоваться для распределения нагрузки.

☆ 25.4. Перечислите четыре основных фактора, влияющих на производительность. Для каждого из них приведите пример приложения, способного легко захватить весь ресурс. Укажите способы решения проблемы в каждом конкретном случае.

☆ 25.5. Найдите в Internet или в `man`-страницах информацию о команде `hdparm`. Какими опциями она располагает для проверки скорости доступа к диску? Как с помощью этой команды можно в некоторых случаях повысить скорость доступа к диску?

☆☆ 25.6. Выберите любые две программы, потребляющие заметный объем системных ресурсов. Воспользуйтесь командой `vmstat` и другими рассмотренными в этой главе средствами для анализа работы каждой программы. Укажите, какие действия программы приводят к повышенному потреблению ресурсов. Подтвердите свои выводы конкретными числами.

Глава 26

Взаимодействие с Windows

Сегодня администраторам очень часто доводится иметь дело со средами, в которых установлены одновременно и Windows-, и Linux-системы. А раз так, они должны знать о том, что существует немало способов, которыми эти операционные системы могут помогать друг другу. Например, Windows-приложения, помимо всего прочего, могут запускаться с рабочего стола Linux или получать доступ к находящимся на сервере Linux принтерам и файлам, а Linux-приложения могут отображать свои пользовательские интерфейсы на рабочем столе Windows.

Обе платформы имеют свои сильные стороны, и их можно заставить работать совместно. Windows — это популярная и удобная настольная система, способная выступать в роли "мостика" между пользователем и выходящим из стены сетевым кабелем. А Linux — это надежная и масштабируемая инфраструктурная платформа. Так что давайте не будем ссориться, ладно?

26.1. Вход в систему Linux из Windows

У пользователей может часто возникать желание поработать в сеансе интерпретатора команд C или `bash`, не вставая из-за своего компьютера Windows. С точки зрения клиента, это проще всего сделать при помощи программы `telnet`, которую Microsoft поставляет вместе с Windows. Однако в наши дни в большинстве систем Linux серверы `telnet` отключены по причинам, связанным с безопасностью. Люди, желающие получить к ним доступ, должны иметь эмулятор терминала, который понимает SSH.

📖 О SSH более подробно рассказывалось в разделе 20.11.

Существует несколько реализаций SSH для Windows. Нашей любимой является простое и эффективное приложение PuTTY, которое распространяется с открытым исходным кодом. Оно поддерживает многие из тех функциональных возможностей, которые мы привыкли видеть в родных терминальных приложениях, таких как `xterm`. Единственной жалобой в адрес приложения PuTTY, которую нам доводилось слышать, было то, что оно не умеет правильно обрабатывать многобайтовые символы; так что пользователям, работающим с азиатскими языками, пожалуй, будет лучше использовать Tera Term Pro вместе с таким подключаемым модулем, как TTSSH, хотя такая установка обеспечивает поддержку только одного протокола SSH. PuTTY можно найти по адресу:

```
www.chiark.greenend.org.uk/~sgtatham/putty
```

а Tera Term Pro — по адресу:

```
hp.vector.co.jp/authors/VA002416/teraterm.html
```

SSH также поддерживает передачу файлов, а PuTYY включает целых две предназначенных для этой цели клиентских программы с интерфейсом командной строки: `psftp` и `pscp`. Заядлым пользователям Windows, которые никогда не имели дела с командной строкой, скорее всего больше придется по душе имеющая графический интерфейс клиентская программа WinSCP, доступная на `winscp.sf.net`.

Еще одним неплохим вариантом является установить более общий пакет UNIX-on-Windows Cywin и запускать доступные в нем SSH-утилиты из `rxvt`. О Cygwin более подробно будет рассказываться чуть позже в этой главе, в разделе 26.4.

Модную, не имеющую пока никаких негативных отзывов Java-реализацию SSH под названием MindTerm предлагает компания AppGate (`www.appgate.com`). Для личного использования MindTerm доступна бесплатно. Запускается MindTerm в любой системе, которая поддерживает Java, и может конфигурироваться различными способами.

Интересной функцией SSH является способность переадресовывать порты TCP туда и обратно между клиентом и сервером. Например, эта функция позволяет администратору настроить на клиенте локальный порт, который будет переадресовывать входящее соединение на другой порт, находящийся на машине, доступ к которой возможен только с сервера. Хотя эта функция открывает множество новых возможностей, она также является потенциально опасной и потому должна учитываться администратором при предоставлении SSH-доступа к своему серверу. К счастью, ее можно отключить на стороне сервера и тем самым ограничить возможности SSH только доступом к терминалу и передачей файлов.

26.2. Получение доступа к удаленным настольным средам

Графические настольные среды на Linux связаны с распространяемой бесплатно системой X Window System, которая не имеет никакого отношения к Microsoft Windows. Система X Window была разработана в исследовательских центрах Массачусетсского технологического института в середине 80-х гг. прошлого столетия и принята в качестве стандарта всеми производителями рабочих станций UNIX и дистрибутивов Linux. Ей довелось пережить несколько серьезных обновлений; стабильная база была окончательно достигнута лишь в версии 11, которая была впервые опубликована в начале 90-х гг. прошлого столетия. Номер версии протокола был добавлен к ее названию: так и полу-

чилось имя "X11", под которым она известна сегодня. (Слово "Windows", используемое само по себе, всегда, как в этой главе, так и в реальном мире, обозначает операционную систему Microsoft Windows.)

X11 — это клиент-серверная система. Сервер X Server отвечает за отображение данных на экране пользователя и за получение входных данных с клавиатуры или мыши пользователя. Он взаимодействует с клиентскими приложениями через сеть. Вовсе не обязательно, чтобы сервер и клиенты запускались на одной и той же машине.

Архитектура X Window более подробно описывалась в главе 22.

Запуск сервера X Server на компьютере Windows

X11 — это богатый протокол, который за свою историю не раз расширялся. Поэтому реализация сервера X Server является довольно сложной. Несмотря на это, сегодня реализации сервера X Server существуют практически для каждой операционной системы. Сама система X Window безразлична к ОС, так что клиенты X11, работающие на Linux-компьютере могут отображать сервер X Server, работающий под управлением Microsoft Windows, и позволять пользователю управлять ими точно так же, как если бы этот пользователь сидел за системной консолью.

К сожалению, первые разработчики протоколов X не уделили должного внимания вопросам безопасности. Каждая программа, которая подключается к серверу X Server, может считывать все, что вводит с клавиатуры обслуживающий этот сервер администратор, и видеть все, что он отображает на своем экране. Еще больше ухудшает дело то, что удаленным программам даже не нужно отображать окно при получении доступа к серверу X Server: они могут просто "тихонько прокрадываться" на заднем плане.

Со временем для защиты X11 было предложено несколько методов, но все они оказались несколько сложными. Вывод таков: лучше не разрешать никаких удаленных подключений к своему серверу X Server, если нет уверенности в том, что делаешь. Большинство серверов X по умолчанию конфигурируется так, чтобы они не допускали удаленных соединений, так что угрозы безопасности быть не должно вплоть до тех пор, пока не будет запущена программа `xhosts` (или ее аналог), предоставляющая разрешение на удаленный доступ.

К сожалению, предоставление прав на удаленный доступ — это единственный путь, когда нужно сделать так, чтобы программы выполнялись в Linux, а их интерфейсы отображались в Windows. Как же запустить удаленное приложение, не предоставив разрешения на удаленный доступ серверу X Server? Наиболее распространенный метод предусматривает использование функциональной возможности SSH-протокола, которая была специально разработана для поддержки X11. В таком случае между клиентами X, работающими на удаленном хосте, и локальным сервером X будет создан безопасный канал. Программы, запускаемые на удаленном хосте, будут автоматически отображаться на локальной машине, но благодаря чудодейственным операциям SSH локальный сервер X будет воспринимать их так, будто бы они были запущены локально.

📖 О протоколе SSH более подробно рассказывалось в разделе 20.11.

Обратите внимание на то, что переадресация X будет работать, только если функции переадресации X включены на обеих сторонах: как на стороне сервера SSH, так и на стороне клиента SSH. Например, если используется SSH-клиент PuTTY в Windows, функцию переадресации X11 можно включить прямо на его экране установки и потом, на стороне сервера SSH (соответственно, на стороне клиента X11, т.е. на компьютере

Linux), удостовериться в том, что файл `/etc/ssh/sshd_config` содержит следующую строку:

```
X11Forwarding yes
```

В случае изменения конфигурации SSH-сервера потребуется перезапустить процесс `sshd` для того, чтобы новая конфигурация вступила в силу.

Если Apple предоставляет бесплатную версию X Server для Mac OS X, то Microsoft, к сожалению, ничего подобного для Windows не предлагает. Однако существует одна бесплатная версия X Server, доступная на сайте проекта Cygwin (`cygwin.com`), которая работает очень хорошо после того, как ее сконфигурировать. К сожалению, ее конфигурирование может оказаться довольно нелегкой задачей для тех, кто не знает, как выполняется конфигурирование X Server на Linux. К коммерческим версиям X Server для Windows относятся eXeed и X-Win32. Эти версии предлагают гораздо более простую процедуру конфигурации, но по довольно высокой цене.

VNC: технология виртуальных сетей

В конце 90-х гг. прошлого столетия несколько человек из исследовательского центра компании AT&T Labs в Кембридже (Великобритания) разработали систему для удаленного доступа к настольным системам под названием *VNC* (Virtual Network Computing — технология виртуальных сетей). Их идеей было "подружить" считающиеся уже примитивными, но зато простые терминалы с современными оконными системами. В отличие от X11, протокол VNC не работает с отдельными приложениями. Вместо этого он создает целый виртуальный рабочий стол (или предоставляет удаленный доступ к существующему) в виде отдельного компонента. Под управлением VNC специальный сервер X11 функционирует на центральной машине, а для получения к нему доступа применяется специальное приложение типа программы просмотра.

AT&T опубликовала программное обеспечение VNC на условиях либеральной лицензии ПО с открытым исходным кодом. Это позволило другим разработчиками подхватить идею и создать дополнительные реализации сервера и программы просмотра, а также улучшить ограниченную пропускную способность протокола. Сегодня VNC-программы просмотра доступны для большинства устройств, имеющих хоть какие-нибудь средства для графического отображения. VNC-серверы для UNIX/Linux и Windows тоже доступны повсеместно.

Реализация VNC-сервера для Linux, по сути, представляет собой эмулятор графического адаптера, который подключается к серверу X.Org X Window. Запуск `vncserver` через учетную запись Linux приводит к созданию нового виртуального рабочего стола, который функционирует на машине Linux в своем собственном, отдельном мире. После этого пользователи могут использовать VNC-программу просмотра и получать доступ к этому рабочему столу дистанционно.

VNC-серверы в мире Windows обычно не создают никакого дополнительного рабочего стола: они просто экспортируют стандартный рабочий стол Windows в том виде, в котором он отображается на экране. Главным приложением для этой технологии является удаленная поддержка.

В наши дни первые создатели VNC уже владеют своей собственной компанией, которая называется *RealVNC* (`realvnc.com`). Авторы проекта Ultr@VNC (`ultravnc.sf.net`) занимаются разработкой очень быстрой и многофункциональной реализации VNC-сервера для Windows, а участники проекта TightVNC (`tightvnc.com`) работают над

улучшением степеней сжатия. Все они обязательно обмениваются информацией между собой, так что функциональные возможности в разных реализациях, скорее всего, все равно будут похожими.

Протокол VNC предусматривает возможность расширения. Все программы просмотра и серверы могут работать друг с другом в любой комбинации: они подбирают наилучший вариант протокола, понятный обеим сторонам. Функциональные возможности, наличие которых зависит о конкретной реализации (такие как возможность передачи файлов), обычно доступны только при использовании сервера и клиента из одного и того же проекта.

Windows RDP: протокол удаленных рабочих столов

Начиная с версии Windows 2000 Server, каждая система Windows имеет техническую возможность для предоставления графического удаленного доступа нескольким пользователям одновременно. Компонент удаленного доступа называется *Remote Desktop* и использует протокол *RDP* (Remote Desktop Protocol — протокол удаленных рабочих столов) для обеспечения связи между клиентом и сервером.

При наличии приличного оборудования ПК на одном сервере Windows Server 2003 запросто можно запускать до 50 одновременных сеансов Microsoft Office. Из-за лицензионных ограничений в Windows XP этой замечательной возможностью можно пользоваться только для одного единственного сеанса: при подключении с удаленной машины локальный сеанс будет автоматически завершен.

Если имеется установка Windows Server, можно активизировать службу Terminal Server и использовать распространяемую с открытым исходным кодом программу `rdesktop` (`www.rdesktop.org`) для получения доступа рабочему столу Windows прямо с рабочей станции Linux. (Клиенты для Windows, Mac OS и Windows CE тоже доступны непосредственно на сайте Microsoft.) Как и протокол VNC, протокол RDP не поддерживает никаких состояний. Это, соответственно, позволяет снова подключаться к прерванному сеансу, а также и осуществлять удаленное обслуживание.

RDP даже разрешает пересылать удаленному сеансу Windows ресурсы локального принтера и диска. Протестируйте установку и все сами увидите — главное не внедряйте ее до тех пор, пока не разберетесь с лицензией. Служба Terminal Server предоставляет 30-дневную лицензию на тестирование любому, кто к ней подключается, но через 30 дней она больше не будет принимать никаких соединений с этого клиента до тех пор, пока не будет приобретена соответствующая лицензия Terminal Server Client Access License (Лицензия на клиентский доступ к службе Terminal Server).

Вся эта концепция получения удаленного доступа к Windows, похоже, вызывает массу сомнений у сотрудников отдела лицензирования как в Microsoft, так и в других компаниях, занимающихся производством ПО для Windows. Условия лицензий постоянно меняются, потому что там пытаются “отделить свои технологии от чужих”, поэтому всегда дожидайтесь официального подтверждения текущих расценок и только потом сообщайте о новых перспективах своим пользователям. Схемой лицензирования, которую бы хотелось увидеть нам всем (лицензирование по количеству одновременно подключающихся пользователей), похоже, пока никто не занимается. На текущий момент выбирать можно между покупкой лицензии для каждого компьютера и покупкой лицензии для каждого пользователя, который собирается хоть когда-нибудь получать доступ к серверу.

26.3. Запуск Windows и Windows-приложений

Коммерческий продукт VMware (www.vmware.com) позволяет запускать на оборудовании ПК сразу несколько операционных систем одновременно. VMware эмулирует все виртуальные "гостевые" машины поверх базовой операционной системы, в роли которой выступает либо Linux, либо Windows. Какой бы не была базовая операционная система, на виртуальных машинах VMware может быть установлена любая Intel-совместимая операционная система.

Благодаря этому, программа VMware является идеальным средством для разработки и тестирования переносимого программного обеспечения. Она даже поставляется с функциональным средством, которое позволяет ей объединять все имеющиеся виртуальные операционные системы в "сеть", так чтобы они могли взаимодействовать друг с другом и совместно использовать настоящее, доступное в системе соединение Ethernet.

Система Wine (www.winehq.com) поступает по-другому. Она реализует в среде Linux API-интерфейс программирования Windows, тем самым позволяя запускать Window-приложения прямо поверх Linux и X. Эта доступная бесплатно программа транслирует "родные" вызовы API-интерфейса Windows в их Linux-аналоги и может делать это, не используя никакого кода компании Microsoft. Кроме того, она поддерживает протоколы TCP/IP, устройства последовательного доступа и звуковые платы.

Проект Wine был начат еще в 1993 г. Бобом Амстадтом (Bob Amstadt). С тех пор он прошел немалый путь. Сегодня многие приложения Windows запускаются без проблем, а некоторые приложения можно заставить работать лишь при помощи нескольких определенных уловок; подробнее об этом можно узнать на официальном Web-сайте проекта Wine. К сожалению, заставить приложения выполняться под Wine часто не так-то просто. Талантливые разработчики на codeweavers.com написали небольшую коммерческую программу-инсталлятор, которая умеет вынуждать "упрямые" приложения Windows работать так, как надо.

Если выбранное приложение поддерживается CodeWeavers — замечательно. Если нет, нужно дать ему шанс, а вдруг оно сможет приятно удивить. Но если приложение не хочет работать само, и никакие заготовленные подсказки отыскать не удается, будьте готовы потратить внушительное количество времени на привидение его в нужную форму, если уж решили делать это своими силами. При наличии бюджетных средств можете попробовать договориться о помощи с CodeWeavers.

Коммерческой альтернативой пакету Wine является программа Win4Lin компании NeTraverse. Она работает стабильнее, чем Wine, и поддерживает больше приложений Microsoft. Но для того чтобы работать с нею, необходимо настраивать ядро, чего не требует пакет Wine. Программа Win4Lin доступна на Web-сайте www.netraverse.com.

Двойная загрузка и почему ею не стоит пользоваться

Тем, кто хоть когда-нибудь устанавливал Linux на компьютер, на котором ранее была установлена Windows, несомненно, предлагалось создать конфигурацию с двойной загрузкой. В наши дни такие конфигурации функционируют во многом так, как обещают. Сегодня даже стало возможно монтировать Windows-разделы под управлением Linux и получать доступ к файловой системе Linux под управлением Windows. О том, как именно можно настроить конфигурацию с двойной загрузкой, подробно рассказывалось в конце раздела 2.3.

Но не стоит спешить! Тем, кто действительно работает и нуждается в доступе как к Windows, так и к Linux, следует очень скептически относиться к конфигурации с двойной загрузкой. Такие конфигурации представляют собой воплощение закона Мэрфи в его самом худшем виде: они, кажется, почти всегда загружают не ту ОС, и из-за этого даже самая простая операция обычно заканчивается выполнением нескольких повторных загрузок. Компьютеры сегодня стоят так дешево и удаленный доступ настраивается настолько легко, что как правило нет никакого смысла подвергать себя таким бесконечным мучениям.

Альтернатива: OpenOffice.org

Несколько лет назад компания Sun выпустила новую версию пакета StarOffice, который является аналогом пакета Microsoft Office. Эта версия тоже распространяется с открытым исходным кодом и называется OpenOffice.org. В пакет OpenOffice.org входят основные офисные утилиты, такие как редактор электронных таблиц, текстовый редактор, пакет средств для создания презентаций и приложение для рисования. Эти программы могут читать и записывать файлы, сгенерированные их Microsoft-аналогами. Загрузить пакет OpenOffice.org можно по адресу:

```
openoffice.org
```

Пакет OpenOffice доступен на всех основных платформах, включая Windows, Linux, Solaris и многие другие версии UNIX. Также существует версия для Mac OS X. Если нужен пакет с контрактом о поддержке, можно приобрести пакет StarOffice, который отличается от OpenOffice.org только наличием более мощного механизма проверки правописания и базы данных.

26.4. Использование утилит командной строки в Windows

Чего многим пользователям Linux не хватает при работе на системах Windows, так это их любимой утилиты `xterm`, не просто любого старого терминального приложения или "рухляди", называемой модулем DOS, а настоящей программы `xterm`, поддерживающей изменение размеров, цветовые схемы, управление мышью и все модные управляющие последовательности.

Хотя автономного (т.е. не зависящего от X) родного порта `xterm` для Windows не существует, имеется небольшая аккуратная программа `rxvt`, которая очень близка к тому, чтобы стать таковым. Она является частью системы Cygwin, которую можно загрузить с сайта `cygwin.com`. Если вы установите сервер X Server, поставляемый с системой Cygwin, вы сможете пользоваться настоящей программой `xterm`.

Эта система распространяется на условиях общедоступной лицензии GNU General Public License и включает довольно обширный дополнительный набор общих команд UNIX, а также переносимую библиотеку, которая реализует API-интерфейсы POSIX под Windows. Ее подход к согласованию обозначений командной строки и файловой системы UNIX и Windows довольно хорошо продуман и умудряется наделять родные команды Windows многими удобными возможностями интерпретатора команд UNIX. Cygwin не только позволяет пользователям UNIX чувствовать себя в Windows "как дома", но и упрощает запуск программного обеспечения UNIX под Windows. Подробнее о Cygwin можно узнать на сайте `cygwin.com`.

Пакет MKS Toolkit представляет собой коммерческий аналог Cygwin. Более подробную информацию о нем можно найти на сайте MKS, который находится по адресу mkssoftware.com.

Также список программ UNIX, которые могут сами запускаться на Windows, постоянно растет; на текущий момент в него уже входят Apache, Perl, BIND, PHP, MySQL, Vim, Emacs, Gimp, Wireshark и Python.

26.5. Совместимость Windows со стандартами электронной почты и Web

В идеальном мире все бы пользовались для связи открытыми стандартами и были бы счастливы. Но наш мир не идеален, и многие ругают Windows за то, что та представляет собой кучу патентованных протоколов и нарушенных реализаций Internet-стандартов. Пожалуй, они частично правы, но, тем не менее, существует много областей, благодаря которым Windows вполне может нормально функционировать в мире стандартов. Этими областями являются службы электронной почты и Web.

За всю богатую историю Web многие корпорации пытались охватить и расширить Web-возможности таким образом, который бы позволил им защититься от конкуренции и увеличить свои доходы во много раз. Microsoft по-прежнему ведет такую борьбу на уровне браузера с его многочисленными расширениями, подходящими исключительно для Internet Explorer. На базовом уровне протокола HTTP, однако, Windows и Windows-браузеры практически не зависят от платформы.

Microsoft предоставляет свой собственный Web-сервер — IIS, но его производительность всегда была хуже производительности сервера Apache, запускаемого на Linux, причем намного. Тем, кто не "подсел" на серверную технологию типа ASP, использовать Windows-машины в качестве Web-серверов вовсе не обязательно. Как Web-клиенты, однако, эти машины смогут охотно "проглатывать" все Web-содержимое, которое серверы UNIX и Linux будут им подавать.

Что касается электронной почты, Microsoft усиленно предлагает в качестве предпочтительной серверной технологии свой продукт Exchange Server. По-правде сказать, возможности Exchange Server действительно "обходят" возможности почтовых систем Internet-стандарта, особенно когда почтовые клиенты представляют собой компьютеры Windows, запускающие Microsoft Outlook. И это еще не все: Exchange Server также может использовать протокол SMTP для входящей и исходящей почты и передавать почту Linux-клиентам при помощи стандартных протоколов IMAP и POP. На стороне клиента Outlook и ее бесплатный аналог Outlook Express могут подключаться к серверам IMAP и POP. Экспериментируйте, пробуйте любые комбинации и выбирайте наиболее подходящую из них. О POP и IMAP более подробно рассказывалось в конце раздела 18.3.

26.6. Совместное использование файлов при помощи Самба и CIFS

В начале 80-х гг. прошлого столетия компания IBM разработала API-интерфейс, который позволял компьютерам в одной и той же подсети общаться друг с другом, используя не замысловатые номера, а имена. Этот интерфейс получил название Network Basic Input/Output System (Сетевая базовая система ввода-вывода), сокращенно — NetBIOS.

А его расширенная версия, включающая исходный базовый сетевой протокол транспортного уровня, была названа NetBIOS Extended User Interface (Расширенный пользовательский интерфейс NetBIOS), сокращенно — NetBEUI. API-интерфейс NetBIOS стал довольно популярным и потому был адаптирован для использования поверх ряда различных сетевых протоколов, таких как IPX, DECNet и TCP/IP.

Microsoft и Intel разработали на базе NetBIOS протокол совместного использования файлов и назвали его "основным протоколом" (core protocol). Позже он был переименован в протокол Server Message Block (Блок сообщений сервера) или, сокращенно, SMB. Далее протокол SMB эволюционировал в систему CIFS (Common Internet File System — Общая межсетевая файловая система), которая, по сути, представляла собой почищенную и настроенную для работы в глобальных сетях версию SMB. Эта система CIFS и по сей день остается главной технологией для совместного использования файлов на компьютерах Windows.

В мире Windows файловая система или каталог, который делается доступным через сеть, называется совместно используемым ресурсом. Этот термин звучит немного странно для пользователей UNIX, но при описании файловых систем CIFS мы будем применять именно его.

Samba: сервер CIFS для UNIX

Чрезвычайно популярный пакет Samba распространяется на условиях открытой лицензии GNU и реализует серверную часть CIFS в Linux. Он был создан австралийцем Эндрю Триджеллом (Andrew Tridgell), который путем "обратного" проектирования воссоздал код протокола SMB и опубликовал полученный результат в 1992 г.

Сегодня пакет Samba хорошо поддерживается и постоянно активно совершенствуется для расширения его функциональных возможностей. Он предоставляет стабильный, надежный механизм для интеграции компьютеров, работающих под управлением Windows, в сеть Linux. Прелесть пакета заключается в том, что устанавливать его нужно только на сервере — на Windows-компьютере никакого дополнительного программного обеспечения не требуется[1].

CIFS предоставляет пять основных услуг:

- совместное использование файлов;
- сетевая печать;
- аутентификация и авторизация;
- преобразование имен;
- анонсирование служб ("обзор" файловых серверов и принтеров).

Пакет Samba не только позволяет обслуживать файлы через CIFS, но также реализует все основные функции первичного контроллера домена Windows[2]. Поддерживается и ряд расширенных возможностей, а именно: доменные учетные записи Windows, копирование по сети профилей пользователей Windows и спулинг печати.

Большая часть функций пакета Samba реализована в двух демонах: `smbd` и `nmbd`. Первый предоставляет услуги печати и доступа к файлам, а также функции аутентифи-

[1] При условии, что тот уже сконфигурирован на работу в сети.

[2] В сочетании с LDAP Samba также может выступать в роли резервного контроллера домена. Этот вопрос хорошо освещен в книге Джона Терпстра (John Terpstra) *Samba-3 by Example*. А о LDAP более подробно рассказывалось в главе 17.

кации и авторизации, а второй управляет другими важными компонентами CIFS: преобразованием имен и анонсированием служб.

В отличие от файловой системы NFS, которая тесно связана с ядром, пакет Samba не требует модификации ядра и запускается исключительно как пользовательский процесс. Он устанавливает соединения с сокетами, через которые посылаются CIFS-запросы, и ждет клиентских запросов на доступ к ресурсам. Как только запрос поступил и был аутентифицирован, демон `smbd` создает новый экземпляр самого себя, который работает от имени пользователя, сделавшего запрос. В результате все права доступа к Linux-файлам (включая групповые права) остаются ненарушенными. Единственная дополнительная функциональная возможность, которую демон `smbd` реализует, — это сервис блокировки файлов, позволяющий клиентским персональным компьютерам придерживаться привычной для них семантики блокировок.

Инсталляция пакета Samba

Пакет Samba входит в состав всех рассматриваемых в этой книге дистрибутивов Linux. Файлы "заплат", документация и вспомогательные средства доступны на Web-сайте `www.samba.org`. Обязательно следует убедиться в том, что используются самые новейшие из доступных для данной системы пакеты Samba, потому что дефекты предыдущих версий пакета чреваты потерей данных и возникновением проблем с безопасностью.

Во всех системах придется отредактировать файл `smb.conf` (который должен находиться либо в каталоге `/etc/samba/smb.conf`, либо в каталоге `/etc/smb.conf`) для того, чтобы задать поведение Samba. В этом файле указываются каталоги и принтеры, которые должны использоваться совместно, права доступа к ним и общие рабочие параметры Samba. Кроме того, в состав пакета Samba входит образец файла `smb.conf`, который содержит подробные комментарии и может стать хорошей отправной точкой для задания новой конфигурации. Обратите внимание на то, что после запуска Samba проверяет свой конфигурационный файл каждые несколько секунд и загружает все появившиеся изменения.

Также важно помнить о проблемах безопасности, которые могут возникать при совместном использовании файлов и других ресурсов по сети. Для типичного сайта обеспечить базовый уровень безопасности можно, выполнив два действия.

- Явно указав, каким клиентам разрешается получать доступ к ресурсам Samba. За эту часть конфигурации отвечает находящаяся в файле `smb.conf` конструкция `hosts allow`. Нужно всегда проверять, чтобы в ней содержались только те IP-адреса (или диапазоны адресов), которые должны.
- Заблокировав возможность получения доступа к серверу людьми, не работающими в данной организации. Samba использует шифрование только для аутентификации пароля. При транспортировке данных шифрование не применяется. В зависимости от природы хранящихся на сервере Samba данных, администратор может заблокировать возможность получения доступа к серверу за пределами организации, дабы исключить вероятность случайной загрузки файлов посторонними людьми. Это обычно делается на уровне сетевого брандмауэра; Samba использует UPD-порты с номерами 137–139 и TCP-порты с номерами 137, 139 и 445.

После выхода версии Samba 3 вся хорошая документация стала доступна на сайте `samba.org`.

Samba поставляется с разумными значениями по умолчанию для большинства ее конфигурационных опций, поэтому для большинства сайтов будет вполне хватать небольшого конфигурационного файла. При помощи команды `testparm -v` можно отобразить перечень всех конфигурационных опций Samba и значений, которые для них установлены на текущий момент. Этот перечень будет включать как параметры, заданные в файле `smb.conf`, так и параметры, установленные по умолчанию.

Параметры в файле `smb.conf` лучше не изменять, если только они не отличаются от параметров по умолчанию и существует веская причина их заблокировать. Преимуществом такого подхода является то, что в таком случае при обновлении до более новой версии Samba текущая конфигурация будет автоматически адаптировать параметры, рекомендуемые авторами Samba.

Приняв во внимание вышесказанное, обязательно проверяйте, чтобы была включена функция шифрования паролей:

```
encrypt passwords = true
```

Эта опция отвечает за шифрование паролей, которыми обмениваются Windows-клиенты и Samba-сервер. На текущий момент она является параметром по умолчанию, и серьезной ситуации, в которой ее действительно лучше было бы отключить, не существует.

Функция шифрования вынуждает сервер Samba сохранять специальный, имеющий вид хеш-значения Windows-пароль для каждого пользователя. Windows-пароли работают совершенно не так, как UNIX-пароли, и поэтому использовать их из каталога `/etc/shadow` невозможно.

Samba предоставляет для установки этих паролей специальную утилиту — `smbpasswd`. Например, давайте попробуем добавить пользователя `tobi` и установить для него пароль:

```
$ sudo smbpasswd -a tobi
New SMB password: <пароль>
Retype new SMB password: <пароль>
```

Пользователи тоже могут изменять свои собственные пароли в Samba при помощи `smbpasswd`:

```
$ smbpasswd -r smbserver -U tobi
New SMB password: <пароль>
Retype new SMB password: <пароль>
```

В этом примере на сервере `smbserver` изменяется пароль пользователя `tobi`.

Кодирование имен файлов

Начиная с версии 3.0, все имена файлов в Samba кодируются в формате UTF-8. Тем, у кого сервер использует кодировку UTF-8, очень повезло: у них будет идеальная совместимость[3]. Тем, кто находится в Европе и до сих использует на своем сервере один из режимов кодировки ISO 8859, повезло меньше: у них имена файлов, содержащие специальные символы вроде ä, ö, ü или é, будут выглядеть довольно странно при вводе команды `ls` в каталоге, в котором эти файлы были созданы в Samba с помощью UTF-8.

[3] Проверить, может ли система работать в режиме кодировки UTF-8, можно при помощи команды `echo $LANG`.

Решением является указать Samba применять тот же режим кодировки символов, что используется на сервере:

```
unix charset = ISO8859-15
display charset = ISO8859-15
```

Правильность режима кодировки лучше проверять сразу. В противном случае количество файлов со странно закодированными именами начнет расти и их исправление позже может оказаться довольно нелегкой задачей.

Обзор сетевого окружения

Раньше сеть Windows настраивалась очень легко. Кто-нибудь просто подключал все Windows-компьютеры к сегменту сети Ethernet, и системы начинали видеть друг друга по имени и могли обмениваться данными. Подобное было возможно благодаря самоорганизующейся системе с возможностью широковещательной рассылки и голосования: один Windows-компьютер выбирался для выполнения роли главного сервера и потом использовался для поддержания списка доступных ресурсов.

Когда Windows начала поддерживать TCP/IP, эта система была переконфигурирована на использование UDP-рассылок. Эффект был по сути точно таким же, но теперь главным рабочим компонентом стала IP-подсеть, а не сеть Ethernet. Обнаруживаемые хосты появляются в на рабочем столе в разделе Network Neighborhood (Сетевое окружение) или My Network Places (Моя сеть), в зависимости от конкретной версии Windows. Для улучшения обзора каждый Windows-компьютер может также быть добавлен в рабочую группу.

Samba принимает участие в этой системе посредством демона `nmbd`. Демон `nmbd` может даже становится главным средством обзора и брать на себя организацию всех Windows-систем в локальной подсети. К сожалению, с усложнением сети, т.е. появлением в ней нескольких подсетей и серверов, широковещательных рассылок будет не достаточно для того, чтобы Windows-компьютеры могли видеть друг друга.

Устранить эту проблему можно, установив сервер Samba как сервер WINS (Windows Internet Naming Service — Windows-служба имен Internet) для каждой подсети. Чтобы сделать это, нужно просто указать в файле `smb.conf` параметр

```
wins support = yes
```

и затем связать все эти серверы с опциями `wins server`.

Система просмотра производства Microsoft не работает идеально даже в сетях, состоящих исключительно из Windows-компьютеров. Часто случаются странные ситуации, когда хост не отзывается на свое собственное имя или когда система ни в какую не хочет “видеть” расположенный рядом компьютер, хотя он подключен к сети и работает нормально. Даже еще чаще проблемы возникают в больших сетях с множеством подсетей.

Те, кто работает с современными Windows-клиентами (Windows 2000 и выше), имеет DNS-записи для всех своих хостов и не нуждается в системе обзора Network Neighborhood (Сетевое окружение), могут отключить поддержку NetBIOS на Samba-сервере, если хотят. Вещей, о которых нужно волноваться, станет на одну меньше.

```
[global]
disable netbios = yes
name resolve order = host
```

Без NetBIOS, Windows придется использовать DNS для выяснения IP-адреса Samba-сервера, как и для любого другого ресурса в сети. Если Samba-сервер запускается без

поддержки NetBIOS, отпадает и необходимость в запуске демона `nmbd`, а, значит, его следует удалить из сценария запуска Samba, который содержится в `/etc/init.d`.

При подключении клиента к Samba-серверу, на котором была отключена поддержка NetBIOS, лучше всего вводить полное DNS-имя этого сервера. Например:

```
\\server.example.com.\myshare
```

Обратите внимание на наличие точки в конце имени сервера. Она указывает Windows не пытаться отыскать запрашиваемую машину через NetBIOS, а сразу переходить к DNS.

Аутентификация пользователей

В системах аутентификации Windows клиент не доверяет серверу, т.е. пароль пользователя никогда не передается по сети в виде открытого текста. Вместо этого, в них используется алгоритм типа "запрос-ответ".

Если при входе в систему Windows предоставить имя пользователя и пароль, Windows будет пытаться использовать эту информацию для аутентификации везде, где это необходимо, при помощи соответствующего аутентификационного запроса. Благодаря алгоритму "запрос-ответ", ничего опасного в таком поведении нет, кроме того, что незашифрованный пароль сохраняется где-нибудь в ОЗУ. Здорово то, что если пользователь имеет такое же имя пользователя и пароль на Windows-компьютере, как и на Samba-сервере, Samba не будет явно запрашивать у него пароль при получении им доступа к совместно используемым ресурсам Samba. Вся процедура аутентификации будет выполняться незаметно на заднем плане.

Недостатком подхода с аутентификацией типа "запрос-ответ" является то, что серверу приходится сохранять пароли в формате, эквивалентном открытому тексту. В действительности серверные копии паролей локально шифруются, но это защищает их только от случайного просмотра. Злонамеренный пользователь, получивший доступ к зашифрованным таким образом паролям, сможет применять их для получения доступа к ассоциируемым с ними учетным записям без какого-либо дополнительного "взламывания". Пароли Samba следует защищать даже еще тщательнее, чем файл `/etc/shadow`.

В сложных средах с множеством серверов Samba имеет смысл использовать централизованную службу каталогов, проверяющую, чтобы на всех серверах стоял одинаковый пароль. В качестве аутентификационных служб Samba позволяет использовать службу NIS (Network Information Service — Сетевая информационная служба)[4], службу LDAP (Lightweight Directory Access Protocol — Упрощенный протокол доступа к каталогам) и службу Windows. О NIS и LDAP подробно рассказывалось в главе 17.

Существует два основных способа объединить системы аутентификации Windows и Linux. Первый — это сконфигурировать сервер Samba так, чтобы он выступал в роли первичного контроллера домена Windows NT (а не службы Active Directory). Второй — установить на Windows-клиентах приложение pGina (`pgina.xpasystems.com`). Это хитрое приложение заменяет стандартную систему регистрации Windows на каркас, который поддерживает все стандартные службы аутентификации, включая LDAP и NIS.

[4] Хотя в некоторой документации она до сих пор указывается, в Samba версии 3 поддержки для NIS+ уже нет. Похоже, что никто не хотел заниматься этим кодом, потому эта функциональная возможность была удалена (по крайней мере, пока).

Совместное использование основных файлов

Когда у каждого пользователя имеется свой собственный домашний каталог, имеет смысл сделать доступными для совместного использования домашние каталоги:

```
[homes]
comment = Home Directories (Домашние каталоги)
browseable = no
valid users = %S
writeable = yes
guest ok = no
```

Эта конфигурация, например, позволит пользователю `oetiker` получать доступ к своему домашнему каталогу через путь `\\sambaserver\oetiker` с любой системы Windows.

На некоторых сайтах, стандартные разрешения на домашних каталогах Linux позволяют людям просматривать файлы друг друга. Поскольку Samba ограничивает доступ на основании разрешений, имеющихся у файлов UNIX, пользователи Windows, заходящие через CIFS, тоже смогут просматривать домашние каталоги друг друга. Однако, как показывает опыт, подобное поведение часто смущает пользователей Windows и вызывает у них чувство, будто бы они находятся у всех на виду. Строка `valid users` в показанном выше фрагменте конфигурации указывает Samba предотвращать подключения к домашним каталогам других людей. Не добавляйте ее, если такое поведение вас не устраивает.

Samba обращается к своему волшебному разделу `[homes]` только в крайнем случае. Если в конфигурации для домашнего каталога пользователя явно указан какой-нибудь определенный совместно используемый ресурс, установленные там параметры перекрывают значения, указанные в разделе `[homes]`.

Совместно используемые ресурсы

Samba может отображать списки управления доступом Windows (Access Control Lists, ACL) либо на разрешения файлов, либо на ACL-списки (если лежащая в основе файловая система поддерживает их). На практике концепция ACL-списков часто оказывается слишком сложной для большинства пользователей. Поэтому администраторы обычно просто создают отдельный совместно используемый (сетевой) ресурс для каждой группы пользователей, которые в таковом нуждаются, и конфигурируют сервер Samba так, чтобы он сам заботился о предоставлении нужных разрешений. Дальше, когда пользователь пытается смонтировать этот ресурс, Samba проверяет, является ли он членом соответствующей группы пользователей Linux и затем изменяет его уникальный идентификатор (UID) на идентификатор владельца этого совместно используемого ресурса (т.е., созданного специально для этой цели псевдопользователя). Например:

```
[eng]
comment = Group Share for engineering (Совместно используемый ресурс для инженеров)
; Каждый, кто является членом группы eng, может получать доступ к этому
; совместно используемому ресурсу
; Пользователям придется входить в систему при помощи своей учетной записи Samba
valid users = @eng

; Мы создали специальную пользовательскую учетную запись по имени eng.
; все записываемые в этот каталог файлы будут принадлежать этой учетной записи,
; а также группе eng
force user = eng
```

```
force group = eng
path = /home/eng

; Отключаем все ACL-списки NT, потому что мы не собираемся здесь их использовать
nt acl support = no

; Делаем так, чтобы у всех файлов были приемлемые разрешения
create mask = 0660
force create mask = 0660
security mask = 0000
directory mask = 2770
force directory mask = 2770
directory security mask = 0000

; Обычные параметры для совместного пользования
browseable = no
writeable = yes
guest ok = no
```

Подобного эффекта можно добиться и с помощью такой опции Samba, как `inherit permissions`. Если активизировать эту опцию на совместно используемом ресурсе, все новые файлы и каталоги будут наследовать свои параметры от родительского каталога:

```
[eng]
comment = Group Share for engineering
path = /home/eng
nt acl support = no
browseable = no
writeable = yes
inherit permissions = yes
```

Поскольку теперь Samba будет брать параметры из родительского каталога, важно не забыть правильно установить разрешения на корневом каталоге этого совместно используемого ресурса:

```
$ sudo chmod u=rw,g=rws,o= /home/eng
$ sudo chgrp eng /home/eng
$ sudo chown eng /home/eng
```

Обратите внимание на то, что при такой конфигурации все равно нужно создать псевдопользователя `eng`, который будет выступать в роли владельца данного совместно используемого каталога.

Прозрачная переадресация при помощи MS DFS

MS DES (Microsoft's Distributed File System — Распределенная файловая система производства компании Microsoft) позволяет каталогам внутри совместно используемого ресурса заставлять клиентов незаметно автоматически монтировать другие совместно используемые ресурсы при получении к ним доступа. Для заядлых пользователей UNIX и Linux в этом нет ничего удивительного, но для Windows эта концепция в целом является довольно революционной и неожиданной. Отчасти может именно поэтому она широко и не применяется, даже несмотря на то, что DES поддерживается во всех версиях Windows уже давно, начиная с версии Windows 98.

Приведем пример:

```
[global]
; Включаем поддержку MS DFS для данного сервера Samba
```

```
host msdfs = yes
...
[mydfs]
; Эта строка указывает Samba искать символические ссылки в каталоге
; данного совместно используемого ресурса
msdfs root = yes
path = /home/dfs/mydfs
```

Чтобы настроить автоматическое монтирование, нужно создать соответствующие символические ссылки в каталоге `/home/dfs/mydfs`. Например, показанная ниже команда превращает "каталог" `jump` в ссылку на один из двух каталогов на других серверах. (Обратите внимание на одинарные кавычки. Они необходимы для защиты символов обратной косой черты.)

```
$ sudo ln -s 'msdfs:serverX\shareX,serverY\shareY' jump
```

Если указывается несколько источников (как здесь), Windows будет выбирать какой-нибудь один из них. Т.е., пользователи, получающие доступ к каталогу `\\server\mydfs\jump`, теперь фактически будут считывать файлы либо из источника shareX на сервере serverX, либо из источника shareY на сервере shareY, в зависимости от того, какой из них будет доступен. Если файловые системы экспортируются с разрешениями не только на чтение, но и на запись, обязательно нужно позаботиться о механизме для синхронизации файлов. На роль такового может подойти утилита `rsync`.

Samba также позволяет делать так, чтобы все клиенты, получающие доступ к определенному ресурсу совместного пользования, перенаправлялись на какой-нибудь другой сервер. Windows-сервер такого не позволяет.

```
[myredirect]
msdfs root = yes
msdfs proxy = \\serverZ\shareZ
```

Обратите внимание на то, что DES будет работать только для тех пользователей, которые имеют одинаковое имя пользователя и пароль на всех задействованных серверах.

`smbclient`: простой клиент CIFS

Помимо множества серверных функциональных возможностей в состав пакета Samba также входит удобная программа для пересылки файлов с интерфейсом командной строки, которая называется `smbclient`. Этой программой можно пользоваться для получения прямого доступа к любому серверу Windows или Samba. Например:

```
$ smbclient //redmond/joes -U joe
Password: <пароль>
Doman=[REDMOND] OS=[Windows 5.0] Server=[Windows 2000 LAN Manager]
smb: \>
```

После успешной регистрации на файловом сервере можно приступать к навигации и передачи файлов, используя для этого стандартные команды в стиле `ftp` (такие как `get`, `put`, `cd`, `lcd` и `dir`).

Файловая система `smbfs`

Linux включает прямую клиентскую поддержку для файловой системы SMB/CIFS. Это значит, что в любой файловой системе Linux можно смонтировать CIFS-ресурс совместного пользования, который ядро будет понимать напрямую. Например:

```
# mount -t smbfs -o username=joe //redmond/joes /home/joe/mnt
```

Хотя эта функциональная возможность и является полезной, следует помнить о том, что Windows воспринимает монтируемые сетевые каталоги как устанавливаемые определенным пользователем (отсюда и опция `username=joe` в приведенной выше строке), в то время как Linux воспринимает их больше как принадлежащие системе в целом. Windows-серверы обычно не могут понять концепцию, подразумевающую, что к смонтированному Windows-ресурсу совместного пользования может получать доступ несколько разных пользователей.

С точки зрения Linux-клиента все файлы в смонтированном каталоге принадлежат пользователю, который его смонтировал. Т.е., если сетевой каталог смонтировать от лица пользователя root, тогда все файлы в нем будут принадлежать пользователю root и, следовательно, обычные пользователи не смогут записывать в него файлы на Windows-сервере.

Опции монтирования `uid`, `gid`, `fmask` и `dmask` позволяют настраивать эти параметры так, чтобы права владения и права доступа более точно отражали предусматриваемую для данного ресурса политику. Подробнее об этом поведении можно узнать на странице руководства `mount.smbfs`.

Чтобы позволить пользователям самостоятельно монтировать Windows-ресурс совместного пользования, можно добавить в файл `/etc/fstab` следующую строку:

```
//redmond/joes /home/joe/mnt smbfs
    username=joe,fmask=600,dmask=700,user,noauto 0 0
```

Благодаря указанной здесь опции `user`, пользователи теперь смогут монтировать файловую систему путем выполнения следующей команды:

```
$ mount /home/joe/mnt
```

`mount` будет приглашать пользователя предоставить пароль, прежде чем монтировать совместно используемый ресурс.

Хотя стандартной сетевой файловой службой для UNIX является NFS, в некоторых ситуациях для осуществления обмена файлами между компьютерами UNIX и Linux лучше использовать Samba или CIFS. Например, опасно позволять пользователям монтировать сетевые NFS-каталоги в корпоративных файловых системах со своих персональных ноутбуков[5]. Однако можно безопасно использовать CIFS для предоставления этим ноутбуками доступа к домашним каталогах их владельцев.

📖 Об NFS более подробно рассказывалось в главе 16.

IBM не жалеет денег на то, что сделать доступ к CIFS с Linux настолько “бесшовным”, насколько это возможно, с целью превратить Samba в надежную альтернативу NFS. Первым результатом этих усилий является то, что сервер версии Samba 3.x можно сконфигурировать так, чтобы он отображал “расширения UNIX”. Т.е., если включить эту функцию, можно будет видеть все атрибуты файлов UNIX, права владения и типы файлов даже при монтировании совместно используемого ресурса с помощью Samba.

Помимо изменений на стороне сервера также вносятся изменения и на клиентской стороне: на текущий момент ведется усиленная разработка нового клиентского модуля

[5] Безопасность в NFS обеспечивается из расчета на то, что у пользователя не может быть прав пользователя root на клиентской машине и что ни на клиенте, ни на сервере не может быть соответствующих уникальных идентификаторов (UID). На автономных компьютерах такого обычно не бывает.

файловой системы под названием cifs, который уже может похвастаться наличием более богатого, нежели у традиционных систем smbfs, набора отладочных и конфигурационных средств.

26.7. Совместное использование принтеров при помощи Samba

Самый простой способ обеспечить возможность совместного использования принтеров — добавить в файл smb.conf раздел [printers]: это заставит Samba сделать доступными для совместного пользования все локальные принтеры. Для выполнения своей работы Samba пользуется командами системы печати, но поскольку печать в Linux не очень стандартизирована, может понадобиться указать Samba, какая именно система печати используется на сервере путем установки для опции printing соответствующего значения. Перечень поддерживаемых на текущий момент систем печати доступен на man-странице smb.conf.

```
[printers]
; Где должны сохраняться файлы, прежде чем они будут переданы системе печати?
path = /var/tmp

; Пользоваться принтерами разрешается всем.
guest ok = yes

; Пусть Samba знает, что в данном случае совместно используемым ресурсом
; является принтер.
printable = yes

; Показывать принтеры всем, кто их ищет.
browseable = yes

; Указать Samba, какая подсистема печати используется в системе.
printing = LPRNG
```

Windows-клиенты теперь смогут пользоваться этими принтерами как сетевыми, точно так же как если бы те обслуживались Windows-сервером. Однако существует одна небольшая проблема. Windows-клиенты будут хотеть знать, принтер какого именно типа им достался, и приглашать пользователя выбрать подходящий драйвер принтера. Это чревато поступлением довольно большого количества просьб о помощи от пользователей, которые не знают, как поступать в подобной ситуации. А если какой-то из имеющихся принтеров требует драйвер, не поставляемый с Windows, тогда этих просьб будет даже еще больше.

Более подробно о печати рассказывалось в главе 23.

К счастью, сервер Samba можно сконфигурировать так, чтобы он предоставлял необходимые Windows-драйверы принтеров всем Windows-клиентам. Однако прежде чем это делать, нужно выполнить кое-какие подготовительные операции. Во-первых, следует удостовериться в том, что Samba будет вести себя как сервер печати, путем добавления соответствующих записей в раздел [global] файла smb.conf:

```
[global]
; Кто у нас отвечает за администрирование принтеров
printer admin = printadm

; Следующий параметр по умолчанию всегда имеет указанное справа значение
disable spoolss = no
```

```
; Не пытайтесь отобразить эту информацию; вы все равно не можете добавлять принтеры
show add printer wizard = no

; Если нужно, чтобы выполнять печать могли все
guest ok = yes
browseable = no
```

Теперь Samba знает, что является сервером печати, и будет воспринимать пользователя `printadm` как своего администратора.

Если вы собираетесь предоставлять драйверы принтера для своих Windows-клиентов, тогда у вас должно быть место для хранения этих драйверов. Таким местом может быть специальный сетевой каталог `[print$]`:

```
[print$]
comment = Printer Driver Area
; Место для хранения драйверов принтеров
path = /var/lib/samba/printers
browseable = yes
guest ok = yes
read only = yes

; Кому разрешено администрировать репозиторий драйверов принтеров
write list = printadm
```

Прежде чем начинать загружать драйверы принтеров на новый сервер печати, следует позаботиться о еще нескольких деталях на уровне системы. Во-первых, нужно удостовериться в том, что учетная запись `printadm` существует и имеет права на получение доступа к Samba:

```
$ sudo useradd printadm
$ sudo smbpasswd -a printadm
```

Во-вторых, Samba сможет сохранять драйверы принтеров только при условии, если соответствующая структура каталогов уже существует и ее владельцем является пользователь `printadm` (как указано в опции `write list`):

```
$ sudo mkdir -p /var/lib/samba/printers
$ sudo cd /var/lib/samba/printers
$ sudo mkdir W32X86 WIN40
$ sudo chown -R printadm .
```

Здесь возможны два варианта: либо пойти к Windows-компьютеру и загрузить драйверы принтеров оттуда, либо воспользоваться средствами Samba и сделать это все из командной строки. К сожалению, простого способа узнать, что именно должно устанавливаться для конкретного драйвера, нет, поэтому в большинстве случаев рекомендуется использовать первый подход. Только если предстоит повторно устанавливать драйвер на множестве серверов, имеет смысл изучить процедуру его установки и научиться воспроизводить ее при помощи средств командной строки.

Установка драйвера принтера из Windows

Чтобы установить драйверы принтеров с Windows-клиента, установите соединение с сервером Samba путем ввода в диалоговом окне Run (Запуск программы), появляющемся после выбора в меню Start (Пуск) пункта Run (Выполнить), следующей строки: `\\`*`сервер-samba`*`.example.com`. Windows попросит ввести имя пользователя и пароль для входа на сервер Samba. Введите имя пользователя и пароль учетной записи

printadm. Если все пойдет хорошо, появится окно со списком имеющихся на этом сервере совместно используемых ресурсов.

Отыщите подпапку Printers (Принтеры): в ней должны отображаться принтеры, которые были сделаны доступными для совместного использования. Щелкните правой кнопкой мыши на каком-нибудь пустом месте между пиктограммами принтеров, чтобы отобразить диалоговое окно Server Properties (Свойства принтера), и затем добавьте свои любимые драйверы принтеров через вкладку Drivers (Драйверы).

Загружаемые драйверы будут помещаться в каталог, указанный в [print$]. На этом этапе можно будет быстро заглянуть в свойства того или иного загруженного драйвера. Именно этот список файлов придется предоставлять утилите командной строки Samba, если когда-нибудь возникнет желание автоматизировать процесс загрузки этого драйвера.

Загрузив все необходимые драйверы, следует связать их с конкретными принтерами. Чтобы сделать это, отобразите панель Properties (Свойства) каждого принтера по очереди (щелкая правой кнопкой мыши и выбирая в появляющемся контекстном меню пункт Properties (Свойства)) и выберите подходящий драйвер на вкладке Advanced (Дополнительно). Затем откройте диалоговое окно Printing Defaults (Параметры печати по умолчанию) и измените параметры печати. Даже если указанные параметры печати вас устраивают, все равно внесите хоть какое-нибудь небольшое изменение, чтобы вынудить Windows сохранить структуры данных конфигурации на сервере Samba. Тогда Samba будет предоставлять эти данные получающим доступ к принтеру клиентам. Невыполнение этого последнего шага может закончиться сбоями на клиентах из-за невозможности отыскать подходящую конфигурацию при попытке использования принтера.

Установка драйвера принтера из командной строки

Как многие уже наверняка догадались, некоторые из перечислявшихся шагов довольно трудно воспроизвести, не используя Windows, особенно это касается установки принтерных параметров по умолчанию. Но если на сервере Samba нужно установить сотни принтеров, все-таки имеет смысл попробовать сделать это из командной строки. Такой метод особенно хорошо подходит для PostScript-принтеров, потому что Windows-драйвер для PostScript-принтеров работает правильно и без информации о конфигурации по умолчанию.

Итак, если вы пометили себе, какие файлы требуются для данного драйвера, вы сможете установить этот драйвер из командной строки. Чтобы сделать это, сначала скопируйте требующиеся файлы в каталог [print$]:

```
$ cd ~/mydriver
$ smbclient -U printadm '//samba-server/print$' -c 'mput *.*'
```

Далее назначьте драйвер определенному принтеру. Предположим, что используется простой PostScript-принтер со специальным PPD-файлом:

```
$ rpcclient -U printadm -c "\
adddriver \"Windows NT x86\" \"Our Custom PS:\
PSCRIPT5.DLL:CUSTOM.PPD:PS5UI.DLL:PSCIPT.HLP:NULL:NULL:PSCRIPT.NTF\"" \
samba-server
```

Символы обратной косой черты в конце строк позволяют разбить команду на несколько отдельных строк для ясности; вы можете опустить их и ввести команду в виде одной строки, если хотите. Символы обратной косой черты перед двойными кавычками указывают на наличие вложенных наборов кавычек.

В длинной строке приведенного выше примера содержится информация, отображаемая в диалоговом окне свойств принтерного драйвера, которое доступно, когда драйвер принтера устанавливается из Windows.

- Длинное имя принтера.
- Имя файла драйвера.
- Имя файла данных.
- Имя конфигурационного файла.
- Имя справочного файла.
- Имя языкового монитора (указывайте здесь значение NULL, если языкового монитора нет).
- Тип данных по умолчанию (указывайте здесь значение NULL, если такого типа данных нет).
- Разделенный запятыми список дополнительных файлов.

Сконфигурировать принтер так, чтобы он использовал один из загруженных драйверов, можно выполнив такую команду:

```
$ rpcclient -U printadm -c "\
set driver \"myprinter\" \"Our Custom PS\"" samba-server
```

26.8. Отладка сервера Samba

Обычно сервер Samba работает нормально, не требуя никакого особого внимания. Но если проблема все-таки появилась, попробовать выявить ее причину можно при помощи двух таких основных источников отладочной информации, как журнальные файлы отдельных клиентов и команда `smbstatus`. Для начала нужно удостовериться в наличии соответствующих параметров журнальных файлов в своем конфигурационном файле:

```
[global]
; Комбинация символов %m указывает, что для каждого клиента должен
; записываться отдельный файл.
log file = /var/log/samba.log.%m
max log size = 1000

; Объем подлежащей регистрации информации. Также можно указывать уровень
; регистрации для компонентов системы. (Здесь указывается, что в общем должен
; использоваться уровень 3, но для аутентификации должен быть выбран уровень 10.)
log level = 3 auth:10
```

Чем выше уровень регистрации, тем больше будет объем отладочной информации. На занесение данных в журналы тратиться определенное время, поэтому не стоит требовать слишком большого количества деталей до тех пор, пока они не понадобятся для откладки, поскольку это может очень сильно замедлить работу системы.

Ниже показаны журнальные записи, сгенерированные после одной неуспешной и одной успешной попытки соединения:

```
[2004/09/05 16:29:45, 2] auth/auth.c:check_ntlm_password(312)
 check_ntlm_password: Authentication for user [oetiker] -> [oetiker] FAILED
  with error NT_STATUS_WRONG_PASSWORD
[2004/09/05 16:29:45, 2] smbd/server.c:exit_server(571)
 Closing connections
```

```
[2004/09/05 16:29:57, 2] auth/auth.c:check_ntlm_password(305)
 check_ntlm_password: authentication for user [oetiker] -> [oetiker] ->
  [oetiker] succeeded
[2004/09/05 16:29:57, 1] smbd/service.c:make_connection_snum(648)
 etsuko (127.0.0.1) connect to service oetiker initially as user oetiker
  (uid=1000, gid=1000) (pid 20492)
[2004/09/05 16:29:58, 1] smbd/service.c:close_cnum(837)
 etsuko (127.0.0.1) closed connection to service oetiker
[2004/09/05 16:29:58, 2] smbd/server.c:exit_server(571)
 Closing connections
```

Команда `smbcontrol` позволяет изменять уровень отладки на функционирующем сервере Samba без внесения изменений в файл `smb.conf`. Например:

```
$ sudo smbcontrol smbd debug "4 auth:10"
```

Эта команда установила бы для уровня глобальной отладки уровень 4, а для уровня отладки вопросов, связанных с аутентификацией — уровень 10. Аргумент `smbd` указывает, что такие уровни отладки должны быть установлены для всех демонов `smbd` в системе. Выполнить отладку конкретного установленного соединения можно следующим образом: использовать команду `smbstatus` для выяснения того, какой демон `smbd` обслуживает данное соединение, а затем передать его идентификатор (PID) команде `smbcontrol`, чтобы та осуществила отладку именно этого соединения. При уровнях регистрации выше 100 в журналах начнут появляться (зашифрованные) пароли.

Команда `smbstatus` отображает информацию об активных соединениях и заблокированных файлах. Эта информация может особенно пригодиться при выявлении проблем блокировки (например, для выяснения, кто из пользователей открыл файл `xyz` для чтения/записи в монопольном режиме). В первом разделе ее выходных данных перечисляются все ресурсы, к которым подключался пользователь, а во втором — любые активные блокировки файлов[6].

```
Samba version 3.0.5
PID     Username    Group       Machine
-------------------------------------------------------------------
12636   zauck       ee          zhaka (192.168.1.228)
29857   milas       guests      beshil (192.168.1.123)

Service  pid         machine     Connected at
-------------------------------------------------------------------
milasa   29857       beshil      Fri Sep 3 17:07:39 2004
zaucker  12636       zhaka       Thu Sep 2 12:35:53 2004

Locked files:
Pid         DenyMode       Access  R/W    Oplock Name
-------------------------------------------------------------------
29857       DENY_NONE      0x3     RDWR   NONE   /home/milasa/hello.dba
12636       DENY_NONE      0x2019f RDWR   NONE   /home/zaucker/aufbest.doc
```

Если уничтожить демон `smbd`, ассоциируемый с определенным пользователем, все его блокировки исчезнут. Некоторые приложения умеют справляться с этим и будут просто заново ставить блокировку, если она им нужна. Другие приложения (такие как MS Access) будут зависать или “умирать страшной смертью”, требуя выполнения

[6] В выводе команды `smbstatus` также содержится несколько очень длинных строк; мы убрали их, чтобы сделать код более удобным для восприятия.

множества щелчков на стороне Windows только для того, чтобы их закрыли. И как бы странно это не звучало, но такая процедура все-таки пока еще никогда не приводила к повреждению. Но в любом случае к заявлениям Windows о том, файлы были заблокированы каким-нибудь другим приложением, все-таки следует относиться с осторожностью. Нередко Windows оказывается права, и проблему можно устранить, просто закрыв приложение на клиентской стороне вместо того, чтобы применять к нему "грубую силу" на стороне сервера.

26.9. Рекомендуемая литература

- Terprstra, John H. *Samba-3 by Example: Practical Exercises to Successful Deployment. 2nd Edition*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. (Электронная версия этой книги доступна на сайте `samba.org`.)
- Terprstra, John H. and Jelmer R. Vernooij. *The Official Samba-3 HOWTO and Reference Guide.2nd Edition*. Upper Saddle River, NJ: Prentice Hall PTR, 2006. (Электронная версия этой книги доступна на сайте `samba.org`.)

26.10. Упражнения

26.1. Зачем может понадобиться блокировать доступ через Internet к портам 137–139 и 445 на сервере Samba?

26.2. Установите программное обеспечение Cywin на компьютер с Windows и воспользуйтесь `ssh` в `rxvt` для подключения к компьютеру Linux. Какие изменения появились в программе PuTTY?

☆☆ 26.3. Сравните производительность клиента, получающего доступ к файлам через сервер Samba, и клиента, который обращается к файлам через обычный сервер CIFS (т.е. Windows-компьютер). Если на обоих тестовых серверах установлено разное оборудование, придумайте, как добиться того, чтобы эти различия не влияли на результаты тестирования. (Может потребоваться доступ с правами суперпользователя.)

☆☆ 26.4. Воспользуйтесь анализатором пакетов (типа `tcpdump` или Wireshark) для перехвата `telnet`-сеанса между Windows и Linux. Инсталлируйте программу PuTTY и повторите процедуру. Проанализируйте результаты обоих проверок. (Необходим доступ с правами суперпользователя.)

☆☆ 26.5. Сконфигурируйте сервер печати Samba так, чтобы он использовал драйверы принтеров Windows для всех предоставляемых им для совместного использования принтеров. Удостоверьтесь в том, что принтеры имеют приемлемую конфигурацию по умолчанию.

Глава 27

Устройства последовательного доступа

Стандарт последовательного интерфейса RS-232C, появившийся в 1969 г., позволяет обмениваться данными устройствам множества различных типов, выпускаемым многими компаниями. Этот стандарт — один из немногих компьютерных стандартов интерфейса, который остался практически неизменным на протяжении почти 40-летней истории и все еще используется в наши дни.

Последовательные порты служат для связи с самыми разными устройствами, включая принтеры, терминалы и другие компьютеры. Они применяются даже в узкоспециализированном оборудовании (устройствах смены носителей, температурных датчиках и — кто бы мог подумать! — швейных машинах). Устройство подключают к системе либо непосредственно (с помощью кабеля), либо по телефонной линии с модемами на обоих концах соединения.

В этой главе рассказывается о том, как подключать к системе устройства последовательного доступа и конфигурировать программное обеспечение для работы с ними. В наших примерах чаще всего фигурируют модемы и принтеры; другие устройства подключаются аналогично.

Первые разделы главы посвящены знакомству с устройствами последовательного доступа и соответствующими кабельными системами. Затем, начиная с раздела 7.8, речь пойдет о программном обеспечении, которое традиционно применялось для поддержки аппаратных терминалов. Такие терминалы теперь — редкость, но их “души” живут в Linux в виде псевдотерминалов и оконных систем. Оставшаяся часть главы (начиная с

раздела 27.14) содержит базовые сведения о модемах, отладке устройств последовательного доступа и стандарте USB.

27.1. Разъем RS-232C

Большинство последовательных портов работает в соответствии с одним из вариантов стандарта RS-232C. Этот стандарт определяет электрические характеристики и назначение каждого сигнального провода, а также разводку контактов традиционного 25-контактного последовательного разъемного соединения DB-25 (рис. 27.1).

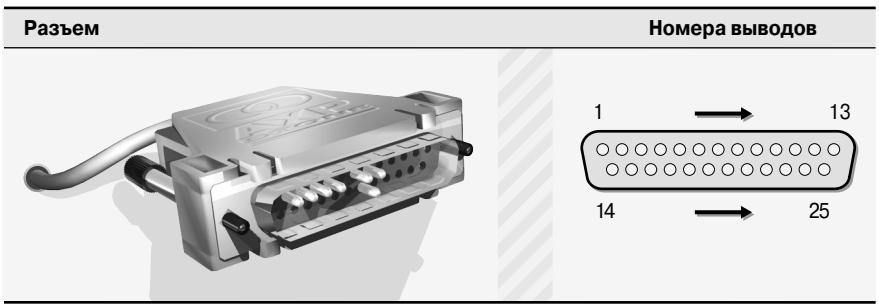


Рис. 27.1. Вилка разъема DB-25

Поскольку стандарт RS-232C[1] предназначен для распространения сигналов, многие из которых не используются в основных режимах передачи данных, в практической работе полный набор сигнальных проводов не используется. Кроме того, разъемные соединения DB-25 слишком громоздки. Поэтому сейчас широко применяются альтернативные модели разъемов (описаны в разделе 27.2).

В традиционном интерфейсе RS-232 использовалась экранированная витая пара (обычно это многожильный провод сортамента 22). В современном же интерфейсе используется та же неэкранированная витая пара, которая является стандартной для последовательных Ethernet-соединений. С технической сточки зрения этот кабель не соответствует спецификации RS-232, но на практике, как правило, его применение не создает никаких проблем.

Изначально в RS-232 применялись сигналы постоянного тока напряжением ±12 В, но сейчас больше распространено напряжение ±5 В. Иногда используется напряжение ±3 В. Чем выше напряжение, тем меньше интерференция сигналов. Все эти показатели соответствуют спецификации RS-232, поэтому допускается соединение устройств с разными уровнями напряжений.

Интерфейс RS-232 не является электрически сбалансированной системой: для передачи данных в каждом направлении используется отдельный проводник. Следовательно, особые электрические свойства витой пары не столь важны при последовательной передаче данных, как, скажем, в Ethernet. Более того, когда два провода передачи данных (TD и RD) скручиваются в одну пару, может произойти снижение надежности и диапазона распространяемых сигналов, так что лучше этого не делать.

[1] С технической точки зрения сегодня правильнее называть его стандартом EIA-232-E. Но если вы будете так говорить, вряд ли вас кто-нибудь поймет.

Нет общепринятого соглашения о том, какие сигналы стандарта RS-232 должны совместно распространяться по витой паре. Некоторые специалисты рекомендуют спаривать провод SG (сигнальная земля) с проводами TD и RD, но, во-первых, это требует выделения дополнительного провода, а во-вторых, появляется несколько путей распространения сигнала "земля". Насколько нам известно, придерживаться подобного соглашения необязательно.

Разъемное соединение DB-25 состоит из вилки (разъем с торчащими штырьками, DB-25P) и розетки, или гнезда ("материнская" часть разъема с соответствующими отверстиями, DB-25S). Возле штырьков и отверстий нанесены крошечные числа от 1 до 25. Лучше всего они видны, если поднести разъем к свету и посмотреть на него под углом. Иногда маркируются только выводы 1, 13, 14 и 25.

На рис. 27.1 была изображена вилка DB-25. Во всех последовательных разъемных соединениях номера контактов на розетке зеркально отражают номера на вилке, чтобы при стыковке разъемов контакты с одинаковыми номерами совпадали. Схема в правой части рисунка соответствует показанной в левой части ориентации вилки.

Обратите внимание: у разъема, изображенного на рисунке, всего семь штырьков. Это наиболее типичный случай. Сигналы интерфейса RS-232 и соответствующие им контакты разъемного соединения DB-25 перечислены в табл. 27.1. На практике используются только сигналы 1–8 и 20, остальные можно игнорировать.

Таблица 27.1. Сигналы интерфейса RS-232 и соответствующие им контакты разъемного соединения DB-25

Контакт	Сигнал	Контакт	Сигнал
1	FG — корпусная земля	14	STD — вторичный сигнал TD
2	TD — передаваемые данные	15	TC — синхронизация передачи
3	RD — принимаемые данные	16	SRD — вторичный сигнал RD
4	RTS — готовность к передаче	17	RC — синхронизация приема
5	CTS — готовность к приему	18	не назначен
6	DSR — готовность данных	19	Вторичный сигнал RTS
7	SG — сигнальная земля	20	DTR — готовность терминала
8	DCD — обнаружение несущей	21	SQ — детектор качества сигнала
9	положительное напряжение	22	RI — индикатор вызова
10	отрицательное напряжение	23	DRS — селектор скорости передачи данных
11	не назначен	24	SCTE — внешняя синхронизация передачи
12	SDCD — вторичный сигнал DCD	25	BUSY — занято
13	SCTS — вторичный сигнал CTS		

Для последовательных устройств существуют две конфигурации кабельной системы: DTE (Data Terminal Equipment — терминальное оборудование) и DCE (Data Communications Equipment — коммуникационное оборудование). Эти конфигурации определяют, какие сигналы устройство будет ожидать на тех или иных контактах разъемного соединения.

Устройства конфигурируются либо в режиме DTE, либо в режиме DCE, хотя некоторые поддерживают оба варианта (но не одновременно). Компьютеры, терминалы и прин-

теры чаще всего относятся к типу DTE, тогда как модемы являются DCE-устройствами. Последовательные порты DTE- и DCE-устройств могут взаимодействовать друг с другом в произвольных сочетаниях, но в каждом конкретном случае требуются разные кабели.

Смысла в одновременном существовании двух конфигураций нет, поскольку для всего оборудования может использоваться одна и та же разводка контактов. Просто это один из многих бессмысленных исторических пережитков стандарта RS-232.

Ниже перечислены особенности обеих конфигураций.

- Разводка контактов в любом разъемном соединении RS-232 всегда одинакова, независимо от того, вилка это или розетка (штырьки всегда совпадают с соответствующими отверстиями) и где находится разъем: на кабеле, DTE- или DCE-устройстве.
- Спецификация RS-232 основана на модели прямого соединения DTE- и DCE-устройств. Под "прямым" соединением понимается подключение линии TD DTE-устройства к линии TD DCE-устройства и т.д., т.е. все одноименные контакты соединяются друг с другом.
- Названия сигналов отражают работу DTE-устройств. Например, название сигнала TD (transmitted data — передаваемые данные) в действительности означает "данные, передаваемые от DTE-устройства к DCE-устройству". Несмотря на это контакт TD служит для *приема* данных на DCE-устройстве. Аналогичным образом, контакт RD является входным на DTE-устройстве и выходным на DCE-устройстве.
- Когда кабелем соединяются два DTE-устройства (компьютер и терминал либо два компьютера), их нужно заставить воспринимать противоположную сторону как DCE-устройство. Например, оба устройства будут предполагать передачу данных по линии TD и прием — по линии RD, поэтому необходимо соединить провода крест-накрест, связав выходной контакт одного устройства с входным контактом другого, и наоборот.
- Подобное "перекрещивание" при соединении двух DTE-устройств требуется для трех групп сигналов: 1) TD и RD, о чем говорилось выше; 2) RTS и CTS; 3) контакт DTR должен быть соединен с контактами DCD и DSR на противоположном конце.
- Кабель, соединяющий два DTE-устройства, называют нуль-модемным. Несмотря на это подключать к нему модемы нельзя, ведь модем — это DCE-устройство! Кабель для модемов называется просто "модемным" или "прямым".

Из-за этой путаницы нередко делаются попытки упорядочить номенклатуру DTE- и DCE-устройств, искусственно внеся различия в разводку контактов (например, путем переименования контакта TD DCE-устройства в RD, и наоборот). Сама кабельная система при этом остается неизменной. Мы советуем игнорировать любые материалы, в которых говорится о DTE-разводке или DCE-разводке; их нельзя считать надежными источниками информации.

Первоначально предполагалось, что DTE-устройства оснащены вилками, а DCE-устройства — розетками. Со временем разработчики аппаратных средств поняли, что вилки чаще подвержены поломкам. Сегодня в дорогостоящей вычислительной технике, как правило, ставят розетки, а кабели с обоих концов имеют вилки[2].

[2] Мы рекомендуем избегать употребления сленговых наименований разъемов. Общепринятыми считаются термины "вилка" и "розетка". Стандартные названия соединителей DB-25P и DB-25S образованы от терминов "pin" — "контакт" и "socket" — "гнездо".

На рис. 27.2 изображены разводки контактов и схемы соединений нуль-модемным и прямым кабелями. Показаны только “полезные” сигналы, используемые на практике.

Рис. 27.2. Разводки контактов и схемы соединения кабелей для разъемов DB-25

27.2. Альтернативные разъемные соединения

В следующих подразделах описываются наиболее распространенные альтернативные разъемные соединения: DIN-8, DB-9 и RJ-45. Несмотря на конструктивные различия, эти соединения обеспечивают передачу тех же электрических сигналов, что и разъем DB-25. Устройства, в которых используются разные разъемы, всегда совместимы, если правильно выбран переходный кабель.

Мини-разъем DIN-8

Мини-разъемы DIN-8 применяются во многих портативных компьютерах и рабочих станциях. Это почти круглые компактные разъемы, которые имеют выводы для семи сигналов стандарта RS-232 (рис. 27.3).

У местных поставщиков компьютеров обычно можно приобрести неразборные кабельные переходники DB-25/DIN-8. Не пытайтесь сделать их самостоятельно, поскольку разъем DIN-8 так мал, что его невозможно собрать вручную. Разводка контактов разъема DIN-8 приведена в табл. 27.2.

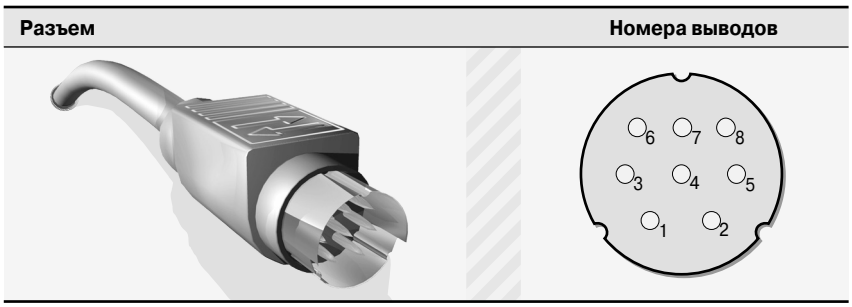


Рис. 27.3. Вилка мини-разъема DIN-8

Таблица 27.2. Разводка контактов для прямого кабельного перехода DIN-8/DB-25

DIN-8	DB-25	Сигнал
3	2	TD — передаваемые данные
5	3	RD — принимаемые данные
6	4	RTS — готовность к передаче
2	5	CTS — готовность к приему
4,8	7	SG — сигнальная земля
7	8	DCD — обнаружение несущей
1	20	DTR — готовность терминала

Разъем DB-9

Девятиконтактный разъем DB-9 (внешне напоминающий уменьшенный DB-25) обычно применяется в персональных компьютерах. Он обеспечивает передачу восьми сигналов стандарта RS-232 (рис. 7.4).

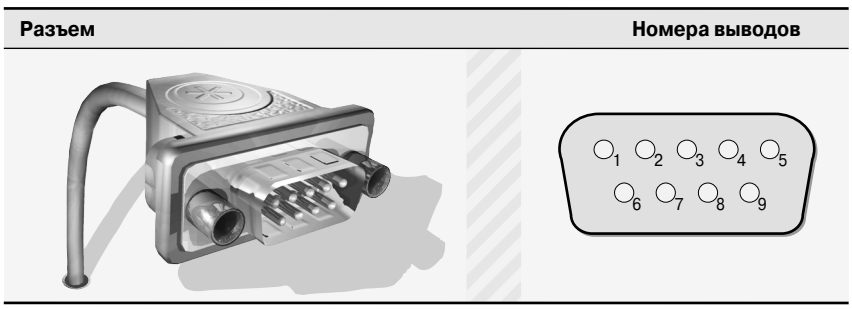


Рис. 7.4. Вилка разъема DB-9

У местных поставщиков персональных компьютеров обычно можно приобрести готовые переходные кабели DB-9/DB-25. Соответствующая разводка контактов представлена в табл. 27.3.

Таблица 27.3. Разводка контактов для прямого кабельного перехода DB-9/DB-25

DB-9	DB-25	Сигнал
3	2	TD — передаваемые данные
2	3	RD — принимаемые данные
7	4	RTS — готовность к передаче
8	5	CTS — готовность к приему
6	6	DSR — готовность данных
5	7	SG — сигнальная земля
1	8	DCD — обнаружение несущей
4	20	DTR — готовность терминала

Разъем RJ-45

RJ-45 — это 8-проводной модульный телефонный разъем. Он напоминает стандартный разъем RJ-11, используемый в телефонных аппаратах в США, но у последнего только 4 штырька[3], а у разъема RJ-45 — 8 (рис. 27.5). Во многих случаях применение разъемов RJ-45 упрощает использование уже проложенных в здании последовательных кабелей (витых пар, предназначенных для соединений Ethernet).

Гнезда RJ-45 практически не встречаются в компьютерах и обычных последовательных интерфейсах, но их часто применяют в качестве промежуточных соединителей при разводке линий последовательной передачи данных через коммутационные панели. Разъемы RJ-45, как правило, используются не с многожильным кабелем витых пар, а с плоским телефонным кабелем. Оба варианта допустимы в последовательных соединениях, но витая пара обеспечивает лучшее качество сигнала при передаче данных на большие расстояния. Плоский телефонный кабель никогда не считался подходящим для Ethernet-соединений, поэтому для снижения риска подключения устройств с помощью неподходящего типа кабеля многие организации применяют только витые пары.

Разъемные соединения RJ-45 компактны, безопасны и дешевы. Они закрепляются на кабеле с помощью специального обжимного инструмента, причем вся процедура длится менее минуты.

Существует несколько стандартов, определяющих соответствие контактов разъема RJ-45 контактам разъема DB-25. В табл. 27.4 описан официальный стандарт RS-232D, который почти не используется.

Другой стандарт — это система Дэйва Йоста (Dave Yost), в которой каждое устройство снабжено гнездом RJ-45 и используются стандартизированные кабели, позволяющие соединять как DCE-, так и DTE-устройства.

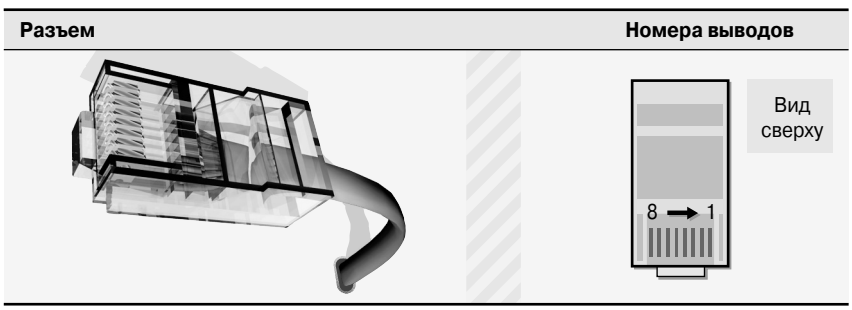


Рис. 27.5. Вилка разъема RJ-45

[3] У разъема RJ-11 имеется шесть пазов для штырьков, но установлено только четыре провода. Настоящий 4-проводной телефонный разъем обозначается RJ-10. Разъем RJ-11, в котором установлены все шесть штырьков, имеет обозначение RJ-12.

Таблица 27.4. Разводка контактов для прямого кабельного перехода RJ-45/DB-25

RJ-45	DB-25	Сигнал
1	6	DSR — готовность данных
2	8	DCD — обнаружение несущей
3	20	DTR — готовность терминала
4	7	SG — сигнальная земля
5	3	RD — принимаемые данные
6	2	TD — передаваемые данные
7	5	CTS — готовность к приему
8	4	RTS — готовность к передаче

Стандарт Йоста для разъемных соединений RJ-45

🕮 Эту спецификацию разработал Дэйв Йост (`Dave@Yost.com`).

Вот особенности схемы, которая позволяет успешно справляться с трудностями стандарта RS-232:

- все кабельные разъемы — одного типа (вилки RJ-45);
- не проводится различий между DTE- и DCE-устройствами;
- необходим соединительный кабель только одного вида;
- можно быстро обрабатывать кабели, используя всего один обжимной инструмент.

Каждый последовательный порт на аппаратном устройстве получает собственный, соответствующим образом разведенный переходник DB-25/RJ-45 или DB-9/RJ-45. Этот переходник наглухо привинчивается к порту. Теперь порты представляют собой один и тот же соединительный разъем — розетку RJ-45, независимо от того, каким был исходный разъем: DB-25 или DB-9, DTE- или DCE-типа, представлял ли он собой вилку или розетку. Более того, отныне все последовательные порты передают и принимают данные через одни и те же контакты.

После установки переходников к последовательному порту можно подключать различные устройства без использования нуль-модемных или нуль-терминальных кабелей, без изменения разводки контактов на кабельных разъемах и без изготовления специальных кабелей. С помощью кабеля одного вида можно соединять модем и компьютер, модем и терминал, терминал и компьютер, терминал и терминал, компьютер и компьютер и т.д.

В разработанной Йостом версии стандарта используется 8-проводной плоский кабель. Выводы разъемов на обоих концах прижимаются к проводам кабеля специальным обжимным инструментом, поэтому паять их не нужно. В связи с существующим предубеждением в отношении этого типа кабелей, в некоторых организациях стандарт Йоста применяют к неэкранированным витым парам.

Имеются три сигнальных провода (один информационный и два управляющих) для каждого направления плюс пара проводов типа “земля”. Кабели разведены не обычным образом (т.е. когда каждый вывод разъема соединяется с соответствующим выводом на другом конце кабеля), а с “перекруткой”, “зеркальным отображением”, “обратной сты-

ковкой" — называйте как угодно. При таком способе разводки кабель соединяет штырек номер 1 на одном конце кабеля со штырьком номер 8 на другом конце и т.д. Эта схема возможна потому, что расположение сигнальных проводов в плоском кабеле симметрично, т.е. для каждого провода, по которому передаются данные, есть провод, по которому данные принимаются, расположенный зеркально к первому относительно оси кабеля[4].

В фабричных кабелях RJ-45 обычно используется прямая разводка, поэтому необходимо снять разъем с одного конца кабеля и вмонтировать новый с обратной разводкой. Увеличивать длину кабеля можно с помощью соединителей типа "розетка-розетка", но помните: два перекрученных кабеля, соединенных таким способом, образуют кабель с прямым соединением.

Переходники DB-25/RJ-45 производят многие фирмы. Их внутренняя цветовая кодировка не совпадает с цветами кабелей. Переходники, соединители и провода можно свободно купить в магазинах электроники, но, к сожалению, нигде не описано их применение при работе с интерфейсом RS-232.

🕮 Дополнительная информация о кабеле категории 5 приведена в разделе 14.2.

Рассматриваемая схема была рассчитана на использование плоского кабеля, в котором все провода расположены в ряд. В противоположность этому в витой паре имеются четыре пары проводов, и в каждой паре провода скручены между собой по всей длине кабеля. Если используется витая пара (например, кабель категории 5), разводка проводов должна быть не такой, как в обычных соединениях RJ-45 (10BaseT, телефонная линия и т.д.). Нужно сделать так, чтобы провода 3:4 и 5:6 образовывали пары. Другое сочетание может привести к возникновению перекрестных помех сигналов данных. Разводка остальных проводов не имеет значения, но обычно спаривают провода 1:2 и 7:8.

Внутри переходника находится гнездо RJ-45, из которого выходят восемь проводов. К этим проводам с помощью обжимного инструмента прикреплены штырьки RS-232 (или, в зависимости от ситуации, контактные гнезда). Штырьки просто вставляются в отверстия разъема RS-232, а затем кожух переходника защелкивается. Разводка контактов разъема приведена в табл. 27.5.

Таблица 27.5. Разводка проводов для переходников Йоста RJ-45/DB-25 и RJ-45/DB-9

		Контакт DCE			Контакт DTE		
Кабель RJ-45	**Переходник**	**DB-25**	**DB-9**	**Сигнал**	**DB-25**	**DB-9**	**Сигнал**
1 Коричневый (к серому)	Голубой	4	7	RTS	5	8	CTS
2 Голубой (к оренжевому)	Оранжевый	20	4	DTR	8	1	DCD
3 Желтый (к черному)	Черный	2	3	TD	3	2	RD
4 Зеленый (к красному)	Красный	7	5	GND	7	5	GND
5 Красный (к зеленому)	Зеленый	7	5	GND	7	5	GND
6 Черный (к желтому)	Желтый	3	2	RD	2	3	TD
7 Оранжевый (к голубому)	Коричневый	8	1	DCD	20	4	DTR
8 Серый (к коричневому)	Белый	5	8	CTS	4	7	RTS

[4] Дэйв не говорит об этом явно, но нужно понимать, что физической "перекрутки" кабелей быть не должно. Поскольку разъемы на противоположных концах направлены в разные стороны, эффект достигается автоматически.

Есть, однако, одна проблема: оба провода "земли" должны быть заведены в одно гнездо DB-25 или DB-9 (контакты 7 и 5, соответственно). Впрочем, их можно обжать с помощью специальных инструментов.

Для некоторых DTE-устройств требуется, чтобы на линии DSR был сигнал, прежде чем они смогут посылать данные. Этот сигнал обычно выдается DCE-устройством, но можно совершить "подлог", сведя вместе контакты 20 и 6 (или 4 и 6 в разъеме DB-9). В этом случае DTE-устройство будет получать сигнал DSR от самого себя при выдаче сигнала DTR.

В DCE-принтерах контакт 7 переходника RJ-45 (коричневый провод) должен быть подключен к линии DSR (контакт 6 в разъемах DB-25 и DB-9). Прочтите документацию к принтеру, чтобы узнать, может ли он посылать сигналы квитирования по линии DSR, а не DCD.

🕮 Дэйв Йост, Лос-Альтос, Калифорния, июль 1999 г.

Не воспринимайте советы, касающиеся линии DSR, как немедленное руководство к действию. Они приведут к желаемому результату, только если конкретное устройство нуждается в соответствующей настройке (или не работает в стандартной конфигурации). Обычные устройства не всегда поддерживают описанные изменения.

27.3. Аппаратная и программная несущие

При подсоединении и включении устройства Linux ожидает, что сигнал обнаружения несущей (DCD) перейдет на высокий уровень (положительное напряжение). Этот сигнал подается на восьмой контакт разъемного соединения DB-25. Если в последовательном кабеле есть линия DCD и компьютер "обращает на нее внимание", значит, используется аппаратная несущая. В большинстве систем допускается также применение программной несущей, когда компьютер "делает вид", что сигнал DCD всегда присутствует.

Для некоторых устройств (в частности, для терминалов) программная несущая — это подарок судьбы. Она позволяет в каждом последовательном соединении обходиться всего тремя линиями: передаваемых данных, принимаемых данных и сигнальной земли. Но в модемных соединениях сигнал DCD обязателен. Если терминал подключен через модем и сигнал обнаружения несущей пропадает, модем должен разрывать соединение (особенно при междугородных звонках). Режим программной несущей для последовательного порта обычно задается в конфигурационном файле (например, `/etc/gettydefs` или `/etc/inittab` в случае регистрационного терминала и `/etc/printcap` в случае принтера) той клиентской программы, которая обслуживает этот порт. Если нужно включить этот режим "на лету", воспользуйтесь командой `stty -clocal`. Например, команда

```
# stty -clocal < /dev/ttyS1
```

включает программную несущую для порта `ttyS1`.

27.4. Аппаратный контроль передачи данных

Назначение сигналов CTS и RTS — обеспечить такую скорость передачи данных, чтобы приемное устройство успевало их обрабатывать. Например, если существует опасность переполнения буфера модема (возможно из-за того, что соединение с удаленным хостом работает медленнее, чем последовательный канал между локальным компьюте-

ром и модемом), модем может приказать компьютеру сделать паузу, пока буфер не освободится.

Контроль передачи данных имеет большое значение для принтеров и высокоскоростных модемов. В системах, где аппаратный контроль передачи отсутствует (из-за того, что последовательные порты не поддерживают этот режим, либо из-за того, что в последовательном кабеле выводы CTS и RTS не подключены), его иногда можно имитировать программным путем с помощью управляющих ASCII-символов XON и XOFF. Однако программный контроль должен явно поддерживаться высокоуровневым программным обеспечением, хотя даже в этом случае его реализация затруднена.

Символам XON и XOFF соответствуют сочетания клавиш <Ctrl+Q> и <Ctrl+S>. Это представляет проблему для пользователей редактора emacs, где при нажатии клавиш <Ctrl+S> по умолчанию вызывается команда поиска. Чтобы исправить проблему, необходимо связать команду поиска с другим сочетанием клавиш или воспользоваться командами `stty start` и `stty stop` для изменения установок драйвера терминала.

Большинство терминалов игнорирует сигналы CTS и RTS. Те немногие терминалы, которые для установления связи требуют, чтобы по этим линиям была проведена процедура квитирования, можно обмануть, соединив перемычкой контакты 4 и 5 на том конце кабеля, который подключен к терминалу. В результате, когда терминал посылает сигнал на вывод 4, заявляя "я готов", то с вывода 5 он получает этот же сигнал обратно, что означает "начинай". Таким же способом можно решить вопрос с квитированием по линиям DTR/DSR/DCD.

Как и в случае программной несущей, параметрами аппаратного контроля передачи можно управлять посредством конфигурационных файлов или команды `stty`.

27.5. Длина кабеля

Стандарт RS-232 определяет, что максимальная длина кабеля при скорости передачи 9600 бит/с должна составлять 75 футов (22,86 м). Но нужно учитывать, что стандарты обычно весьма консервативны, и RS-232 — не исключение. В повседневной работе нам приходилось прокладывать кабель гораздо большей длины (иногда до 1000 футов (305 м)). Как показывает опыт, предел находится где-то между 800 и 1000 футов (244 и 305 м), причем это в значительной степени зависит от конкретных моделей устройств на концах соединения.

Для увеличения максимальной длины линии связи иногда используются повторители. К сожалению, эти устройства часто усиливают только сигналы RD и TD.

27.6. Файлы последовательных устройств

Последовательные порты представляются в системе файлами устройств, расположенными в каталоге `/dev` или его подкаталогах. У большинства компьютеров имеется два встроенных последовательных порта: `/dev/ttyS0` и `/dev/ttyS1`. Дистрибутивы Linux обычно включают полный набор файлов устройств (64 и больше) для дополнительных последовательных портов, но до тех пор, пока в систему не будет добавлено соответствующее оборудование, эти файлы следует игнорировать.

Как всегда, имена файлов устройств не имеют особого значения. Для представления устройств служат старшие и младшие номера устройств, а сами имена файлов являются лишь общепринятым соглашением.

27.7. Команда setserial: установка параметров последовательного порта

Последовательным портам персональных компьютеров можно назначать различные адреса ввода-вывода и номера запросов на прерывание (IRQ). Параметры портов считываются из BIOS при включении питания компьютера. Менять их приходится, только если какое-нибудь "вредное" устройство соглашается работать лишь при выделении ему ресурсов, обычно закрепленных за одним из последовательных портов. К сожалению, драйвер последовательных портов не всегда способен распознать подобные изменения конфигурации без посторонней помощи.

В UNIX эта проблема традиционно решается указанием параметров последовательного порта на этапе компиляции ядра. Слава богу, в Linux этого можно избежать, воспользовавшись командой `setserial`.

При наличии флага `-g` она отображает текущие установки порта:

```
# setserial -g /dev/ttyS0
/dev/ttyS0, UART: 16550A, Port: 0x03f8, IRQ: 4
```

Задавая параметры порта, нужно сначала указать файл устройства, а затем — последовательность названий параметров и их значений. К примеру, команда

```
# setserial /dev/ttyS1 port 0x02f8 irq 3
```

устанавливает адрес ввода-вывода и номер прерывания для порта `ttyS1`. Важно уяснить, что эта команда ни в коей мере не меняет аппаратную конфигурацию системы. Она лишь сообщает конфигурационные параметры драйверу последовательных портов Linux. Физические установки изменяются через BIOS.

Изменения, производимые командой `setserial`, не сохраняются при перезагрузке системы. Стандартного способа сделать их постоянными не существует; в каждом дистрибутиве это делается по-разному.

В Red Hat и Fedora сценарий `/etc/rc.d/rc.sysinit` проверяет наличие сценария `/etc/rc.serial` и, если он существует, выполняет его на этапе начальной загрузки. Готового примера сценария нет, поэтому его придется создать самостоятельно. Просто перечислите команды `setserial`, которые нужно выполнить, по одной на строку. Для надежности можно добавить в начало файла строку `#!/bin/sh` и сделать сам файл исполняемым, хотя это и не обязательно.

В SUSE есть сценарий `/etc/init.d/serial`, управляющий инициализацией последовательного порта. К сожалению, у этого сценария нет конфигурационного файла, поэтому придется отредактировать сам сценарий, внеся в него нужные команды. Сразу предупредим: для формирования команд `setserial` в сценарии применяется собственный метаязык. Правда, в сценарии уже есть множество примеров с комментариями, так что проблем возникнуть не должно.

В Debian есть конфигурационный файл `/etc/serial.conf`, читаемый сценарием `/etc/init.d/setserial`. Пакет `setserial` Ubuntu также содержит этот файл, но он не устанавливается по умолчанию. В этом файле реализовано много расширенных возможностей (например, режим постоянной автоконфигурации), которые, очевидно, используются лишь разработчиками дистрибутива. В простых случаях достаточно удалить символы комментария из нужных строк или добавить требуемые параметры в строку вызова команды `setserial`.

27.8. Конфигурирование программного обеспечения для устройств последовательного доступа

После подключения устройства с помощью соответствующего кабеля необходимо определенным образом сконфигурировать программное обеспечение, чтобы устройство работало эффективно. Перечень конфигурационных задач, которые необходимо решать при подключении нового устройства, зависит от типа устройства и приложений, в которых оно будет использоваться.

- Если это аппаратный терминал, нужно дать системе указание ожидать поступления запросов на регистрацию через порт терминала. Задайте скорость передачи и параметры последовательного соединения. Конфигурация терминалов описана в следующем разделе.
- Модемы, принимающие звонки, конфигурируются аналогично аппаратным терминалам. Тем не менее, в некоторых системах возможны незначительные различия в перечне необходимых процедур.
- Чтобы узнать, как использовать модем при установлении сетевого соединения с удаленным пользователем по протоколу PPP, прочтите раздел 12.13.
- Информация об особенностях настройки принтеров последовательного доступа приведена в главе 23, "Печать". Одни принтеры только принимают данные, а другие могут передавать компьютеру информацию о своем состоянии.
- Специализированное устройство последовательного доступа, которое обслуживается только с помощью собственного программного обеспечения, не требует особой конфигурации. Для доступа к нему достаточно открыть файл устройства. Обратитесь к man-странице команды `termios`, чтобы узнать, как устанавливать скорость передачи, флаги и режим буферизации последовательного порта.

27.9. Конфигурирование аппаратных терминалов

За последние два десятилетия дешевые компьютеры практически полностью вытеснили с рынка текстовые терминалы. Но даже "терминальные" окна, работающие в графическом режиме, используют те же драйверы и файлы конфигурации, что и реальные терминалы, поэтому системные администраторы должны знать, как работают эти древние устройства.

При настройке терминала нужно решить две важные задачи: обеспечить закрепление за терминалом процесса, который будет принимать поступающие от него запросы на регистрацию, и обеспечить доступность информации о терминале после входа пользователя в систему. Но прежде чем приступать к решению этих задач, давайте рассмотрим, как осуществляется регистрация в системе.

Процедура регистрации в системе

📖 О демоне `init` рассказывалось в разделе 2.5.

В ходе регистрации задействуется несколько программ, самая важная из которых — демон `init`. Одна из его задач заключается в порождении процесса, обобщенно называемого `getty`, на каждом терминальном порте, который определен в файле `/etc/`

inittab. Демон getty устанавливает начальные характеристики порта (в частности, скорость передачи и режим контроля четности) и выводит на экран приглашение к регистрации.

Реальное имя демона getty зависит от дистрибутива, а в некоторых дистрибутивах существует даже несколько реализаций демона. В Red Hat, Fedora и SUSE используется упрощенная версия, называемая mingetty, которая обрабатывает запросы, поступающие от виртуальных консолей. Для работы с терминалами и модемами предназначен демон mgetty. В системах Debian и Ubuntu для выполнения всех перечисленных задач служит универсальная версия демона getty, которая под названием agetty используется и в системах SUSE. Более старая реализация, называемая uugetty, в основном вытеснена демоном mgetty.

Чтобы сориентироваться в этом многообразии, рассмотрим утилиты с точки зрения сложности. Проще всех демон mingetty. Он способен лишь обрабатывать запросы, поступающие от виртуальных консолей Linux. На следующем уровне иерархии находится демон agetty, который умеет обрабатывать запросы, поступающие от последовательных портов и модемов. Наконец, на вершине иерархии стоит демон mgetty. Он может обрабатывать не только запросы на регистрацию, но и команды факсов, а также координировать работу модемов, чтобы один и тот же модем мог управлять как входными, так и выходными потоками.

Последовательность событий при регистрации в системе такова:

- демон getty отображает содержимое файла /etc/issue, а также приглашение к регистрации;
- пользователь вводит регистрационное имя в строке приглашения;
- демон getty запускает программу login, передавая ей в качестве аргумента введенное имя;
- программа login запрашивает пароль и сверяет его с записями в файле /etc/shadow [5];
- программа login выводит на экран "сообщение дня", хранящееся в файле /etc/motd, и запускает интерпретатор команд;
- интерпретатор выполняет соответствующие сценарии конфигурации[6];
- интерпретатор отображает на экране приглашение командной строки и переходит в режим ожидания команд.

Когда пользователь выходит из системы, управление возвращается демону init, который "пробуждается" и порождает новый процесс getty для порта терминала.

Большинство параметров процедуры регистрации сосредоточено в файле /etc/inittab, где содержатся команды запуска демонов getty, а также в файле /etc/gettydefs, к которому обращаются некоторые версии демона getty в поисках дополнительной конфигурационной информации.

[5] Если файл скрытых паролей не используется, пароль берется непосредственно из файла /etc/passwd. Кроме того, файл /etc/passwd может заменяться или дополняться административной СУБД, например NIS. Подробнее об этом рассказывается в главе 17.

[6] Файл .profile в случае sh/ksh; файлы .bash_profile и .bashrc в случае bash; файлы .cshrc и .login в случае csh/tcsh.

Файл /etc/inittab

Демон init поддерживает различные "уровни выполнения", которые определяют, какие системные ресурсы задействуются. Существуют семь уровней выполнения, пронумерованных от 0 до 6, плюс уровень S, понимаемый как синоним уровня 1 (однопользовательский режим). При выходе из однопользовательского режима демон init приглашает пользователя ввести номер уровня выполнения, если только в файле /etc/inittab не задан параметр initdefault, как описано ниже. Затем демон просматривает файл inittab в поиске всех строк, соответствующих указанному уровню.

Уровни выполнения обычно устанавливаются таким образом, чтобы у пользователя был один уровень, на котором доступна только консоль, и другой уровень, на котором доступны все терминалы. Можно задавать уровни выполнения так, как того требует конкретная система, но мы рекомендуем не слишком отклоняться от значений по умолчанию.

Записи файла inittab имеют следующий формат:

идентификатор:*уровни_выполнения*:*действие*:*процесс*

Таблица 27.6. Возможные значения поля действие файла /etc/inittab

Значение	Ждать?	Интерпретация
initdefault	–	Задает исходный уровень выполнения
boot	Нет	Процесс запускается при первом чтении файла inittab
bootwait	Да	Процесс запускается при первом чтении файла inittab
ctrlaltdel	Нет	Процесс запускается в ответ на нажатие клавиш <Ctrl+Alt+Del>
once	Нет	Процесс запускается однократно
wait	Да	Процесс запускается однократно
respawn	Нет	Процесс постоянно поддерживается в рабочем состоянии
powerfail	Нет	Процесс запускается при получении демоном init сигнала сбоя питания
powerwait	Да	Процесс запускается при получении демоном init сигнала сбоя питания
sysinit	Да	Процесс запускается перед обращением к консоли

Если одно из значений в поле *уровни_выполнения* совпадает с номером текущего уровня, а значение поля *действие* свидетельствует об актуальности записи, то демон init с помощью интерпретатора sh выполняет команду, заданную в поле *процесс* (или прекращает ее выполнение). В столбце "Ждать?" табл. 27.6 указано, в каких случаях демон init ожидает завершения команды, прежде чем продолжить работу.

В приведенном выше примере в последних двух строках порождаются демоны mingetty для первых двух виртуальных консолей (консоли переключаются нажатием клавиш <ALT+F1> и <Alt+F2>). При добавлении в систему аппаратных терминалов или модемов соответствующие записи файла inittab будут примерно такими же, только в качестве команды должна быть указана строка вызова демона mgetty или getty (agetty в SUSE), поскольку демон mingetty не работает с этими устройствами. Как правило, в качестве значения поля *действие* выбирается respawn, а в поле *уровни_выполнения* указывают 2345.

Команда telinit -q заставляет демон init повторно прочитать файл inittab.

Различные версии демона getty конфигурируются по-разному. Версия getty/agetty, имеющаяся в SUSE, Debian и Ubuntu, немного проще в этом плане, чем mgetty, потому что она берет свою конфигурационную информацию из командной строки (содержащейся в файле /etc/inittab).

Общий формат строки таков:

```
/sbin/getty порт скорость тип_терминала
```

О базе данных terminfo подробнее рассказывалось в разделе 27.9.

где *порт* — это файл устройства для данного порта (имя файла указано относительно каталога /dev); *скорость* — это скорость передачи в бодах (например, 38400), а поле *тип_терминала* определяет стандартный тип терминала, подключаемого к данному порту. Последнее поле является ссылкой на запись базы данных terminfo. Большинство эмуляторов имитирует работу терминала VT100 компании DEC, обозначаемого как VT100. Большинство менее важных параметров в основном связанно с управлением модемами.

Демон mgetty, имеющийся в Red Hat, Fedora и SUSE, обладает более сложными средствами работы с модемами, чем agetty, и к тому же понимает входные и выходные команды факсов. Его конфигурация более рассредоточена. Помимо обычных флагов командной строки демон mgetty может принимать ссылку на файл /etc/gettydefs, в котором указаны детали конфигурации драйвера последовательных портов. Необходимость в этом возникает только при сложной настройке модемов.

В Red Hat информацию о файле gettydefs можно получить с помощью команды man mgettydefs. Соответствующая страница названа так во избежание конфликта со старой man-страницей gettydefs, которая больше не существует ни в одной Linux-системе.

Строка вызова демона mgetty для аппаратного терминала выглядит примерно так:

```
/sbin/mgetty -rs скорость устройство
```

Параметр *скорость* — это скорость передачи в бодах (например, 38400), а *устройство* — это файл устройства для последовательного порта (указывайте полный путь к нему).

Если требуется задать тип терминала по умолчанию, это придется сделать в отдельном файле, /etc/ttytype, а не в командной строке демона mgetty. Формат записей этого файла таков:

```
тип_терминала устройство
```

Здесь параметр *устройство* — это сокращенное имя файла устройства, соответствующего порту, а параметр *тип_терминала* является ссылкой на запись базы данных terminfo. Приведем ряд примеров:

```
linux   tty1
linux   tty2
vt100   ttyS0
vt100   ttyS1
```

Поддержка терминалов: базы данных termcap и terminfo

В Linux поддерживается много разных типов терминалов. Для этого предусмотрена специальная база данных, в которой содержатся характеристики и сведения об особен-

ностях программирования каждой модели терминала. Исторически сложилось так, что существуют две конкурирующие базы данных — termcap и terminfo. Для обеспечения максимальной совместимости в Linux, как правило, присутствуют обе. Первая содержится в файле /etc/termcap, а вторая — в каталоге /usr/share/terminfo. Эти базы данных весьма схожи друг с другом и обычно обозначают типы терминалов одинаковыми названиями, поэтому различия между ними можно считать несущественными.

Обе базы данных содержат сведения о сотнях различных терминалов. В нынешнюю "бестерминальную" эпоху почти все записи являются пережитком прошлого и совершенно не нужны. Достаточно запомнить, что в большинстве случаев эмулируется терминал DEC VT100, если не указано иное. Многие эмуляторы поддерживают также режим "ansi". Типы "linux" и "xterm" используются консолями Linux и программой xterm (терминал X Window), соответственно.

📖 Подробнее о конфигурировании терминалов на этапе регистрации в системе рассказывается в разделе 27.12.

Для того чтобы определить, какой тип терминала используется, Linux-программы проверяют значение переменной среды TERM. Затем проводится поиск дополнительной информации об этом терминале в базе данных termcap или terminfo. Система, как правило, устанавливает переменную TERM в процессе регистрации пользователя, в соответствии с аргументами командной строки демона getty/agetty или содержимого файла /etc/ttytype.

27.10. Специальные символы и драйвер терминала

Драйвер терминала поддерживает несколько специальных функций, доступ к которым осуществляется нажатием особых комбинаций клавиш (как правило, в эти комбинации входит клавиша <Ctrl>). Привязку функций к клавишам можно задать с помощью команд tset и stty. Некоторые из этих функций и обозначения соответствующих им клавиш перечислены в табл. 27.7.

Таблица 27.7. Специальные символы и функции драйвера терминала

Название	Стандартная комбинация	Функция
erase	<Ctrl+?>	Стирает один введенный символ
werase	<Ctrl+W>	Стирает одно введенное слово
kill	<Ctrl+U>	Стирает всю строку
eof	<Ctrl+D>	Посылает терминалу признак конца файла
intr	<Ctrl+C>	Прерывает выполнение текущего процесса
quit	<Ctrl+\>	Уничтожает текущий процесс, создавая дамп памяти
stop	<Ctrl+S>	Останавливает вывод на экран
start	<Ctrl+Q>	Возобновляет вывод на экран
susp	<Ctrl+Z>	Приостанавливает текущий процесс
lnext	<Ctrl+V>	Игнорирует специальное значение следующего символа

По умолчанию версии ядра Linux, предназначенные для персональных компьютеров, генерируют символ удаления (<Ctrl+?>) при нажатии клавиши забоя (она помечается надписью “backspace” или “delete” либо просто стрелкой влево). Раньше во многих UNIX-системах в аналогичной ситуации генерировался другой символ (<Ctrl+H>). К сожалению, существование двух различных стандартов порождает многочисленные проблемы.

Можно воспользоваться командой `stty erase`, чтобы сообщить драйверу терминала о том, какой символ на самом деле используется. Но некоторые программы (например, текстовые редакторы и командные интерпретаторы с режимом редактирования командной строки) имеют “собственное представление” о том, каким должен быть символ забоя, поэтому они не всегда обращают внимание на установки драйвера терминала. Часть программ поддерживает оба варианта символа. Может даже оказаться, что в локальной системе принята одна установка, а в удаленной системе, в которой вы зарегистрировались по сети, — другая.

Устранение этих незначительных, но раздражающих противоречий может потребовать затраты некоторого дополнительного времени (например, в воскресенье). Для каждой программы придется определить собственные правила интерпретации управляющих символов. Полезную информацию по решению этой задачи можно найти в разделе *Linux Backspace/Delete mini-HOWTO* на сайте `www.tldp.org` и в статье Анне Баретта (Anne Baretta) на Web-странице по адресу `www.ibb.net/~anne/keyboard.html`.

27.11. Команда `stty`: задание параметров терминала

Команда `stty` позволяет непосредственно изменять и запрашивать значения различных параметров драйвера терминала. Существует множество опций, большинство которых можно проигнорировать. Обычно названия опций команды `stty` совпадают с теми, которые указаны на `man`-странице `termios`, хотя бывают и расхождения.

Опции команды `stty` могут появляться в любом порядке и в любых сочетаниях. Дефис перед опцией отменяет ее. Например, следующая команда настраивает терминал на скорость 9600 бит/с с контролем по четности и без встроенной поддержки табуляции:

```
$ stty 9600 even -tabs
```

Вот хорошее сочетание опций для простого терминала:

```
$ stty intr ^C kill ^U erase ^? -tabs
```

Здесь опция `-tabs` запрещает драйверу терминала задействовать встроенный механизм табуляции, поскольку многие эмуляторы не поддерживают его в полной мере. Остальные опции назначают специальным символам прерывания (`intr`), удаления строки (`kill`) и удаления символа (`erase`), соответственно, сочетания клавиш <Ctrl+C>, <Ctrl+U> и <Ctrl+?>.

Команду `stty` можно использовать для анализа текущих режимов драйвера терминала и их включения. Команда `stty` без аргументов выдает следующую информацию:

```
$ stty
speed 38400 baud; line = 0;
-brkint -imaxbel
```

Более детальный отчет можно получить с помощью команды `stty -a`:

```
$ stty -a
speed 38400 baud; rows 50; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

Формат вывода остался прежним, но отображена вся имеющаяся информация. О назначении параметров в большинстве случаев можно догадаться интуитивно.

Команда `stty` работает с файловым дескриптором своего стандартного входного потока, поэтому с помощью оператора переадресации, предусмотренного в интерпретаторе команд (<), можно устанавливать и запрашивать режимы не только текущего, но и других терминалов. Изменять режимы чужих терминалов имеет право только пользователь `root`.

27.12. Команда `tset`: автоматическое задание параметров терминала

Команда `tset` переводит драйвер терминала в режим, соответствующий типу терминала. Тип терминала можно задать в командной строке. Если он не указан, используется значение переменой среды `TERM`.

Синтаксис команды `tset` предусматривает возможность изменения значений переменной среды `TERM`. Это удобно, если вы часто регистрируетесь в системе через модем или коммутатор данных и хотите, чтобы драйвер был сконфигурирован в расчете на реальный терминал, который используется на другом конце соединения, а не на нечто общее и бесполезное вроде “коммутируемого терминала” (тип “dialup”).

Предположим, к примеру, что на домашнем компьютере используется программа `xterm`, а система, в которой вы регистрируетесь по модему, воспринимает тип терминала модема как “dialup”. Тогда, если добавить команду

```
tset -m dialup:xterm
```

в файл `.login` или `.profile`, драйвер терминала всегда будет настроен на работу с программой `xterm`.

К сожалению, команда `tset` не так проста, как кажется. Чтобы заставить ее не только задавать режим работы терминала, но и менять значения переменных среды, нужно воспользоваться следующим мини-сценарием:

```
set noglob
eval `tset -s -Q -m dialup:xterm`
unset noglob
```

Здесь подавляется обычный вывод команды `tset` (флаг `-Q`), а вместо этого запрашивается вывод команд интерпретатора, устанавливающих значения переменных среды (флаг `-s`). Поскольку присутствуют обратные кавычки, командные строки, отображаемые командой `tset`, перехватываются и подаются на вход интерпретатора благодаря встроенной команде `eval`. В результате команды будут выполнены так, как если бы их ввел сам пользователь.

Команда `set noglob` запрещает интерпретатору раскрывать метасимволы, такие как "`*`" и "`?`", в выводе команды `tset`, а команда `unset noglob` восстанавливает нормальный режим работы интерпретатора. Это не нужно в интерпретаторах `sh`/`ksh`, поскольку обычно они не раскрывают специальные символы, которые заключены в обратные кавычки. Вид самой команды `tset` не зависит от интерпретатора: проверяя значение переменной среды `SHELL`, она определяет, какие команды следует генерировать.

27.13. Устранение зависания терминала

Некоторые программы (например, редактор `vi`) в процессе работы вносят серьезные изменения в параметры драйвера терминала. Пользователь, как правило, этого не замечает, поскольку при завершении или приостановке программы состояние терминала тщательно восстанавливается. Однако существует опасность, что программа завершит работу аварийно или ее выполнение будет прекращено до завершения процедуры восстановления. Если это случится, терминал может начать вести себя очень странно: перестать правильно обрабатывать символы новой строки, отображать вводимые символы и адекватно выполнять команды.

Еще один способ нарушить работу терминала — применить к бинарному файлу команду `cat` или `more`. В большинстве таких файлов содержится причудливая смесь управляющих символов, которая наверняка окажется "смертельной" для недостаточно устойчивых эмуляторов.

Чтобы решить эту проблему, можно воспользоваться командой `reset` или `stty sane`. Первая из них является всего лишь ссылкой на команду `tset` и может принимать большинство ее аргументов, хотя, как правило, вызывается без аргументов. Обе команды — и `reset`, и `stty sane` — восстанавливают работоспособность драйвера терминала и посылают терминалу соответствующий код сброса, который берется из базы данных `termcap`/`terminfo`, если таковая имеется.

Во многих случаях сброс необходим потому, что терминал был оставлен в особом состоянии, когда вводимые пользователем символы не обрабатываются. В этом состоянии большинство терминалов при нажатии клавиши <Return> или <Enter> генерирует только символ возврата каретки (<Ctrl+M>), а символ новой строки, при получении которого текущая команда посылается на выполнение, не генерируется. Чтобы ввести символ новой строки непосредственно, вместо клавиши <Return> нажмите <Ctrl+J> или клавишу перевода строки (если таковая имеется).

27.14. Модемы

Модем преобразует цифровой последовательный сигнал компьютера в аналоговый, пригодный для передачи по стандартной телефонной линии. К счастью, появление скоростных устройств доступа к Internet превращает эти когда-то важные устройства в первоочередных кандидатов на помещение в музей.

Внешние модемы имеют гнездо RJ-11 для приема аналоговых сигналов и разъем стандарта RS-232 (обычно розетку DB-25) — для цифровых сигналов. На передней панели модема, как правило, имеются лампочки, сигнализирующие о текущем состоянии модема и происходящих в нем процессах. Модем должен устанавливаться в области прямой видимости, чтобы лампочки были хорошо заметны.

Внутренние модемы обычно встречаются только в персональных компьютерах. Плата модема вставляется в слот ISA, PCI или PCMCIA и имеет гнездо RJ-11, которое выводится на заднюю панель компьютера. Внутренние модемы дешевле внешних, но их труднее конфигурировать, и у них нет индикаторных лампочек.

Если вы собрались покупать внутренний модем, проверьте, поддерживается ли он в Linux. Появление быстродействующих процессоров позволило упростить аппаратную часть модемов, поручив обработку некоторых сигналов процессору. К сожалению, для такого рода встроенных модемов (обобщенно называемых Winmodems) требуются специализированные драйверы, которые не всегда поддерживаются в Linux. Информацию по этому вопросу можно найти в документе *Linmodem HOWTO* (доступен на Web-сайте `www.linuxdoc.org`). Полезным источником является также документ *Modem HOWTO*, в котором освещен более широкий круг вопросов, связанных с управлением модемами.

Внутренние модемы обычно конфигурируют так, что с точки зрения пользовательских программ они будто бы подключаются через фантомный последовательный порт. Подобное соглашение позволяет изолировать логические функции модема от его аппаратной реализации. Стандартные программы работают с модемом, не располагая сведениями о его особенностях.

Модемы различаются по общей надежности, но, не имея опыта, эту характеристику оценить трудно. Раньше одни модемы были устойчивей других к помехам в линии. Сегодня в большинстве моделей стоит стандартный набор микросхем от одного из основных производителей, поэтому различия между модемами не столь велики, как прежде.

Высокоскоростные модемы требуют наличия сложных микропрограмм, а в этих программах иногда встречаются ошибки. Производители стараются использовать один и тот же микропрограммный код в разных моделях, вследствие чего и хорошие, и плохие программы распространяются на целые семейства модемов. По этой причине мы рекомендуем покупать модемы широко известных марок.

Протоколы модуляции, коррекции ошибок и сжатия данных

В былые времена имело большое значение, какие именно протоколы поддерживаются модемом, поскольку стандарты постоянно менялись, а производители модемов не всегда реализовывали полный набор протоколов. Сегодня в модемах используются одни и те же стандарты. Единственное различие между моделями модемов — качество микропрограммного кода, электронных компонентов и технической поддержки.

Скорость передачи в бодах определяется частотой, с которой модулируется сигнал несущей. Если имеется более двух уровней сигнала, то за один переход можно передавать несколько битов информации, следовательно, скорость в битах будет выше скорости в бодах. Исторически сложилось так, что скорость передачи данных и скорость передачи сигналов модемами были одинаковыми, что привело к непреднамеренному слиянию единиц измерения “бод” и “бит/с”.

Современные модемы работают по стандарту V.90 “56K”, который в действительности не соответствует пропускной способности 56 Кбит/с. В лучшем случае он обеспечивает скорость связи 33,6 Кбит/с от компьютера к провайдеру и 53 Кбит/с в обратном направлении. Недавно появившаяся усовершенствованная версия стандарта V.90, V.92, обеспечивает скорость связи к провайдеру, равную 48 Кбит/с, а не 33,6 Кбит/с, как ранее. Стандарты V.90 и V.92 позволяют вплотную приблизиться к теоретическому

и практическому пределу скорости распространения сигналов по обычным голосовым телефонным линиям, поэтому в ближайшее время замены им не предвидится.

Помехи в линии передачи могут стать причиной возникновения значительного количества ошибок в модемном соединении. Для борьбы с подобным явлением разработаны различные протоколы коррекции ошибок, которые обеспечивают формирование пакетов передаваемых данных и исправление ошибок на основании проверки контрольных сумм. Эти протоколы защищают пользователей и прикладные программы от помех в линиях связи и обычно не требуют дополнительного конфигурирования.

Алгоритмы сжатия данных позволяют уменьшить количество битов, передаваемых между аналоговыми оконечными устройствами. Коэффициент сжатия варьируется от единицы и меньше (когда передаются данные, уже подвергшиеся сжатию) до 4:1. Типичный показатель — 1,5:1. Как правило, при включении режима сжатия скорость передачи данных увеличивается.

Программа `minicom`: выходные звонки

Традиционные утилиты набора номера `tip` и `cu` относительно непопулярны в Linux, хотя обе они были адаптированы (утилита `cu` обычно входит в состав пакета UUCP, представляющего собой устаревшую систему телефонной связи). Чаще всего в Linux применяются универсальные программы `kermit` и `minicom`, которые обеспечивают эмуляцию терминала и поддерживают протоколы передачи данных. Для нерегулярного использования мы рекомендуем вторую программу, так как она часто включается в состав дистрибутивов. В Debian программа `minicom` не инсталлируется по умолчанию; чтобы ее установить, нужно выполнить команду `apt-get install minicom`.

Как бы там ни было, программа `minicom` имеет больше общего с программами для ПК, чем большинство программ Linux. Тем, кто еще помнит особенности удаленного доступа на скорости 1200 бод посредством терминала MS-DOS, широко использовавшегося в 1986 г., программа `minicom` покажется очень знакомой. Для ее настройки выполните команду `minicom -s` от имени суперпользователя и перейдите в меню “serial port setup”. Задайте файл устройства, включите аппаратный контроль передачи, установите режим кодирования 8N1 (8 бит данных, без контроля четности, один стоповый бит) и убедитесь, что выбрана правильная скорость передачи. Вернитесь в основное меню и выберите команду “save settings as dfl”, чтобы сохранить изменения.

Те, кто знаком со стандартными командами модема, могут вводить их напрямую (например, команда ATDT5551212 означает тоновый набор номера 555-1212). Чтобы это делала программа `minicom`, нажмите <Ctrl+A>, а потом — <D> для перехода в меню набора номера.

Дуплексные модемы

Во многих случаях удобно конфигурировать один модем как на прием, так и на передачу данных, особенно если он реализует функции факса. Для этого требуется особая настройка последовательного порта, поскольку демон `getty` не может просто захватить порт и заблокировать все остальные процессы. Механизм совместного управления портом реализуется с помощью параметров функции открытия системы и включает в себя использование как специальных возможностей драйвера, так и функций ядра. Все программы, совместно работающие с портом, должны следовать определенному протоколу.

Раньше конфигурирование дуплексного модема было сложной задачей, требующей изменения множества системных настроек, а результат не всегда оказывался удовлетворительным. К счастью, большинство приложений Linux хорошо взаимодействует друг с другом, если используется программа `mgetty` или ее предшественница `uugetty`. В целом предпочтительнее первая из них, так как по умолчанию она включает режим совместного доступа к порту.

27.15. Отладка последовательной линии

Наладить работу последовательной линии несложно. Приведем перечень типичных ошибок. Итак, вы:

- забыли дать демону `init` указание повторно прочитать свои конфигурационные файлы;
- забыли установить режим программной несущей при использовании трехпроводных кабелей;
- используете несоответствующий нуль-модемный кабель;
- при пайке или обжиме проводов перевернули вверх ногами разъем DB-25;
- подключили устройство не к тому проводу из-за того, что монтажные схемы оказались неправильными или неточными;
- неверно задали параметры терминала.

Коммутационный тестер — незаменимое средство устранения проблем в кабельных системах. Он подключается к последовательной линии и показывает наличие сигналов в каждом из проводов кабеля. У качественных коммутационных тестеров на каждой стороне есть и вилки, и розетки, поэтому они могут подключаться самыми разными способами. С каждым из представляющих интерес контактов (контакты 2, 3, 4, 5, 6, 8 и 20) связаны индикаторы, которые подсвечиваются, когда контакт активен.

Одни коммутационные тестеры позволяют только наблюдать за сигналами, другие дают возможность осуществлять повторную разводку соединений и подтверждают наличие напряжения на контакте. Например, если есть подозрение, что кабель нужно сделать нуль-модемным, то можно с помощью коммутационного тестера изменить имеющуюся разводку и поменять местами контакты 2 и 3, а также 6 и 20.

🕮 Контактная информация компании Black Box приведена в разделе 14.14.

Лучше не иметь никакого коммутационного тестера, чем иметь плохой. Мы предпочитаем устройство BOB-CAT-B компании Black Box (`www.blackbox.com`).

27.16. Другие распространенные порты ввода–вывода

Раньше последовательные порты были непреложным стандартом для подключения низкоскоростных периферийных устройств к UNIX-системам. Но в настоящее время большинство персональных компьютеров оснащено для этой цели портами USB (Universal Serial Bus — универсальная последовательная шина).

По предоставляемым возможностям USB-порт значительно превосходит последовательные (да и параллельные) порты. Он обеспечивает высокую скорость передачи данных (до 480 Мбит/с), конструктивно прост и использует простые и дешевые стандар-

тизованные кабели. USB-интерфейс быстро становится стандартным для большинства внешних устройств.

В крайнем случае, всегда можно приобрести переходники USB/RS-232C. Их использование позволяет не беспокоиться по поводу подключения устаревших последовательных устройств к новым ПК, которые не имеют встроенных последовательных портов. В частности, хорошо себя зарекомендовали переходники USB/RS-232C, продаваемые компанией Keyspan (`www.keyspan.com`). Драйверы для устройств компании Keyspan входят в состав ядра Linux версий 2.4.22 и последующих.

Параллельные порты были популярны в 80-х гг. и они используют ту же концепцию, что и последовательные порты, но передают за раз восемь битов, а не один. Параллельные интерфейсы традиционно применяются в принтерах, хотя в Windows они использовались также для подключения Zip-дисководов и накопителей на магнитной ленте. Однако для этой цели уже давно стали применять USB-интерфейсы, в связи с чем современные версии Linux поддерживают параллельные порты только для принтеров.

Интерфейс USB

📖 Дополнительную информацию о стандарте USB можно найти на Web-сайте `www.usb.org`.

USB — это универсальный стандарт подключения периферийных устройств, разработанный компаниями Compaq, DEC, IBM, Intel, Microsoft, NEC и Northern Telecom. Первая спецификация шины USB была опубликована в 1996 г. В течение последних десяти лет стандарт USB стал широко использоваться в Windows-системах. В результате все новые компьютеры теперь оснащаются USB-портами, а большинство периферийных устройств выпускается в USB-вариантах.

USB — это отличный интерфейс, и мы полагаем, что он просуществует много лет. Вот его основные характеристики:

- чрезвычайная дешевизна;
- возможность подключать до 127 устройств;
- в USB-кабеле всего четыре провода: питание, “земля” и два сигнальных провода;
- разъемные соединения стандартизированы;
- разъемные соединения невелики по размеру, а сами кабели тонкие и гибкие;
- устройства можно подсоединять и отсоединять без выключения питания;
- скорость передачи данных доходит до 480 Мбит/с (в устройствах по стандарту USB 2.0);
- через специальные переходники к USB-порту можно подключать старые устройства последовательного и параллельного доступа.

Интерфейс USB можно даже применять в локальных сетях, хотя он не предназначен для этого.

Современные версии Linux обеспечивают надежную всеобъемлющую поддержку USB-устройств. В спецификации USB определены стандартные интерфейсы для нескольких классов устройств (в частности, мыши, модемы и запоминающие устройства сверхбольшой емкости), поэтому работать с ними относительно несложно. С другой стороны, для таких устройств, как видеокамеры и сканеры, могут потребоваться специализированные драйверы.

Информацию о доступных USB-устройствах можно найти в каталоге `/proc/bus/usb`. Нумерованные файлы (например, 001, который представляет USB-контроллер компьютера) соответствуют отдельным USB-устройствам. В файле `/proc/bus/usb/devices` содержится подробная информация о характеристиках текущих устройств, а в файле `/proc/bus/usb/drivers` — имена зарегистрированных на данный момент USB-драйверов (независимо от того, используются они или нет). Оба эти файла можно просматривать с помощью программы `less` или любого текстового редактора.

Файловая система `/proc/bus/usb` поддерживает автоматическое конфигурирование устройств и удобна при отладке, но обычно она не используется драйверами ядра напрямую. Обращение к большинству USB-устройств осуществляется посредством традиционных файлов устройств, находящихся в каталоге `/dev`.

Чаще всего USB-интерфейс используется устройствами хранения данных, такими как модули флэш-памяти. Монтирование USB-дисков было подробно описано в разделе 7.9. Постоянно обновляемую информацию о других типах устройств можно найти в перечне устройств, помещенном на сайте `www.linux-usb.org`.

27.17. Упражнения

27.1. Что такое нуль-модемный кабель? Как он используется при подключении DCE- и DTE-устройств?

27.2. Можно ли использовать трехпроводный последовательный кабель для подключения модема? А для последовательного принтера? Почему?

27.3. Как осуществляется аппаратный контроль передачи данных традиционными устройствами последовательного доступа? Что можно сделать, если система не поддерживает аппаратный контроль?

27.4. Что такое псевдотерминал? Какие программы работают с псевдотерминалами?

27.5. Создайте в файле `/etc/inittab` записи, которые:

а) запускают программу `server-fallback` и дожидаются ее завершения, а затем немедленно останавливают работу системы, если поступает сигнал сбоя питания;

б) перезапускают серверную программу `unstable-srv` в случае сбоя;

в) запускают сценарий `clean-temp`, который удаляет все временные файлы при каждом перезапуске системы.

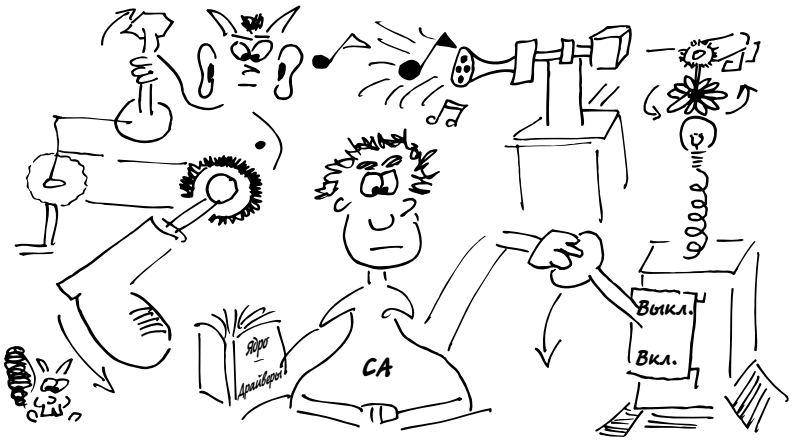
27.6 Предположим, что к компьютеру было подключено новое USB-устройство. Как убедиться в том, что оно распознано и доступно?

☆ 27.7 Сравните стандарты RS-232 и USB.

☆ 27.8. Ваш друг забыл выйти из Linux-системы, оставив ее включенной на ночь в лаборатории. На следующий день он начал сталкиваться со странными проблемами при запуске программ в интерпретаторе команд. Программы внезапно завершались или зависали, а введенные символы исчезали при вводе определенных команд или символов. В то же время, некоторые программы работали вполне корректно. Чем могут быть вызваны подобные аномалии? Как это проверить? Как устранить проблему? Кто мог так “пошутить”?

Глава 28

Драйверы и ядро



Ядро — это часть операционной системы, которая организует доступ к аппаратным средствам через абстрактный высокоуровневый программный интерфейс. Ядро отвечает за реализацию многих концепций, которые пользователями и программами пользовательского уровня принимаются как нечто само собой разумеющееся. В частности, на базе низкоуровневых аппаратных возможностей ядро реализует следующие элементы операционной системы Linux:

- процессы (защита адресных пространств, разделение времени и ресурсов процессора);
- сигналы и семафоры;
- виртуальную память (подкачка, страничный обмен, отображение виртуальных адресов в физической памяти);
- файловую систему (файлы, каталоги, пространство имен);
- общий ввод-вывод (специализированное оборудование, клавиатура, мышь);
- межзадачное взаимодействие (каналы и сетевые соединения).

Ядро содержит драйверы устройств, которые организуют взаимодействие с отдельными элементами аппаратного уровня; остальная часть ядра в основном не зависит от внешних устройств. Взаимосвязь ядра и драйверов устройств аналогична связи между ядром и процессами пользовательского уровня. Когда процесс просит ядро “прочесть первые 64 байта файла `/etc/passwd`”, оно транслирует эту просьбу в команду драйвера устройства, например: “выбрать блок 3348 из устройства 3”. Драйвер представляет эту

команду в виде последовательности двоичных кодов, которые записываются в управляющие регистры устройства.

Ядро написано преимущественно на языке C, хотя в некоторых случаях используется язык ассемблера, что позволяет вызывать специализированные функции устройств, недоступные через обычные директивы компилятора.

Одно из преимуществ Linux заключается в том, что доступность исходного кода ядра позволяет легко, если это слово здесь вообще применимо, писать собственные драйверы устройств и модули ядра. На заре Linux умение писать такие программные компоненты было необходимым, поскольку для эффективного администрирования операционной системы требовалось адаптировать ее к конкретной аппаратной среде.

Сегодня администраторам не приходится копаться в коде ядра. Считается, что подобной деятельностью должны заниматься программисты, а администраторам следует сосредоточивать усилия на организации работы пользователей. Администраторы могут настраивать ядро и добавлять существующие модули, как описано в данной главе, но им не обязательно проходить курс программирования на языке C или языках ассемблера.

Администрирование современных Linux-систем даже не стоит сравнивать с "вспашкой целины", каким оно являлось всего несколько лет назад.

28.1. Адаптация ядра

Linux-системы существуют в мире, где есть десятки тысяч различных компонентов компьютерного аппаратного обеспечения. Ядро должно адаптироваться к любым аппаратным средствам компьютера, на котором установлена система.

Ядро может узнать о системном оборудовании различными способами. Самый простой — предоставление ядру явной информации об устройствах, которые необходимо найти (или в отношении которых можно сделать вид, будто их не существует, как иногда бывает). Некоторые ядра самостоятельно ищут устройства либо на этапе начальной загрузки, либо динамически в процессе работы системы. Последний метод наиболее часто применяется для современных устройств, подобных подключаемым к универсальной последовательной шине (USB), в том числе для карт памяти, модемов, цифровых камер и принтеров. К счастью, Linux предоставляет поддержку широкого множества таких устройств.

В персональных компьютерах, где операционная система Linux особенно популярна, задача создания точного перечня имеющихся аппаратных компонентов чрезвычайно сложна (а иногда вообще нерешаема). Мир персональных компьютеров развивался по законам эволюции, не так уж сильно отличающимся от законов нашего мира, в котором от одноклеточных организмов произошли все существующие формы жизни — от морских ежей до тушканчиков. Ситуация ухудшается тем, что производители персональных компьютеров обычно предоставляют недостаточно технической информации о продаваемых системах, поэтому часто приходится разбирать компьютер, чтобы узнать, например, какой набор сетевых микросхем Ethernet установлен на материнской плате.

Для выживания современным Linux-системам необходима "смешанная диета" из статических и динамических компонентов ядра, что в основном объясняется ограничениями аппаратной среды персональных компьютеров. Иногда системному администратору приходится играть роль доброго доктора, создавая новую конфигурацию ядра.

28.2. Драйверы и файлы устройств

Драйвер устройства — это программа, которая организует взаимодействие системы с определенным аппаратным компонентом. Драйвер играет роль переводчика команд конкретного устройства на "язык" API-функций ядра. Благодаря наличию драйверов обеспечивается приемлемый уровень независимости Linux от внешних устройств.

Драйверы устройств являются компонентами ядра, а не пользовательскими процессами. Тем не менее, доступ к драйверу возможен как из ядра, так и со стороны команд пользовательского уровня. Для последних создаются специальные файлы устройств, хранящиеся в каталоге `/dev`. Ядро преобразует операции, выполняемые над этими файлами, в вызовы функций драйвера.

Персональные компьютеры всегда служили источником хаоса в мире системных администраторов. Существуют многочисленные "стандарты" и всевозможные аппаратные устройства с различными уровнями поддержки со стороны операционной системы. И вот что мы видим:

- в настоящее время существует более 30 различных наборов микросхем SCSI для персональных компьютеров, и все они продаются множеством поставщиков;
- существует более 200 сетевых интерфейсов для персональных компьютеров, каждый из которых предлагается различными поставщиками под разными названиями;
- все время разрабатываются и выпускаются новые, улучшенные, более дешевые устройства, и для каждого из них нужен свой драйвер, работающий с определенной версией Linux.

Учитывая темпы появления новых устройств, практически невозможно выпускать основные дистрибутивы операционных систем, в которых интегрирована поддержка всех новинок. Поэтому нередко приходится самостоятельно добавлять к ядру новые драйверы[1].

В Linux можно инсталлировать только те драйверы, которые разработаны специально для использования в Linux (как правило, еще и для конкретной версии ядра Linux). Драйверы для других операционных систем (например, Windows) работать не будут. Об этом следует помнить, покупая новое оборудование. Кроме того, различные устройства в различной степени совместимы с Linux и при работе в ее среде могут предоставлять различные функциональные возможности. Поэтому, выбирая оборудование, следует учитывать опыт его применения в других организациях.

Изготовители оборудования обращают все больше внимания на рынок UNIX и Linux и часто создают Linux-версии драйверов. Если повезет, к устройству будут прилагаться и драйвер, и инструкции по его инсталляции. Но с наибольшей вероятностью нужный драйвер можно найти на каком-нибудь неофициальном Web-сайте. Как бы то ни было, ниже мы опишем, что происходит в системе при добавлении драйвера устройства.

Файлы и номера устройств

Для многих устройств в каталоге `/dev` имеются соответствующие специальные файлы; исключение в современных операционных системах составляют лишь сетевые устройства. Сложные серверы могут поддерживать сотни различных устройств.

[1] Для просмотра устройств, которые подключены к шине PCI персонального компьютера и обнаружены ядром операционной системы, можно воспользоваться командой `lspci`.

С каждым из файлов связаны старший и младший номера устройства. Посредством этих номеров ядро преобразует обращения к файлу устройства в вызовы нужного драйвера.

Старший номер устройства обозначает драйвер, за которым закреплен данный файл (другими словами, он определяет тип устройства). Младший номер устройства обычно указывает на то, к какому конкретно устройству данного типа следует обращаться. Младший номер иногда называют номером модуля.

Узнать номера устройств позволяет команда `ls -l`:

```
$ ls -l /dev/sda
brw-rw---- 1 root     disk      8,    0 Jan 5 2005 /dev/sda
```

В этом примере выдана информация о первом SCSI-диске. Его старший номер равен 8, а младший — 0.

Младший номер иногда используется драйвером для выбора характеристики устройства. Например, накопителю на магнитной ленте может соответствовать несколько файлов в каталоге `/dev`, обеспечивающих реализацию различных сочетаний параметров, таких как плотность записи и наличие режима обратной перемотки. Драйвер волен интерпретировать младший номер устройства по своему усмотрению. Особенности работы драйвера можно узнать на соответствующей `man`-странице.

Файлы устройств бывают двух типов: блочно-ориентированные (блочные) и байт-ориентированные (символьные). Чтение из блочного устройства и запись в него осуществляются по одному блоку за раз (блок — это группа байтов, размер которой обычно кратен 512), тогда как чтение из символьного устройства и запись в него происходят в побайтовом режиме.

Определенные системные функции удобно реализовывать, используя драйвер устройства, даже если связанное с ним устройство не существует. Такие фантомные устройства называют *псевдоустройствами*. Например, пользователю, зарегистрировавшемуся в системе по сети, назначается псевдотерминал (PTY), который с точки зрения высокоуровневого программного обеспечения ничем не отличается от обычного последовательного порта. Благодаря такому подходу программы, написанные в эпоху алфавитно-цифровых терминалов, могут функционировать в мире окон и сетей.

Когда программа выполняет операцию над файлом устройства, ядро автоматически перехватывает обращение к файлу, ищет в таблице переходов соответствующую функцию и передает ей управление. Для выполнения необычных операций, не имеющих прямых аналогов в модели файловой системы (например, выталкивание дискеты из дисковода), используется системный вызов `ioctl`, который передает пользовательскую команду непосредственно драйверу.

Создание файлов устройств

Файлы устройств можно создавать вручную с помощью команды `mknod`, которая имеет следующий синтаксис:

```
mknod имя_файла тип старший младший
```

где *`имя_файла`* — это создаваемый файл устройства, *`тип`* — тип устройства (`c` — в случае символьного устройства и `b` — в случае блочного), а *`старший`* и *`младший`* — соответственно, старший и младший номера устройства. Если вручную создаете файл устройства, драйвер которого уже имеется в ядре, просмотрите соответствующую этому драйверу `man`-страницу и найдите указанные в ней старший и младший номера.

Исторически так сложилось, что файлы устройств в каталоге /dev создавались администраторами вручную. Большинство систем предоставляло сценарий MAKEDEV, хранящийся в каталоге /dev, который облегчал выполнение этой задачи. Иногда, но не всегда, сценарий MAKEDEV знал, как создать нужные файлы устройств для конкретного аппаратного компонента. Но даже в лучшем случае этот процесс был весьма трудоемким.

Демон udev, поставляемый с ядром Linux, начиная с версии 2.6[2], динамически управляет созданием и удалением файлов устройств в соответствии с фактическим присутствием (или отсутствием) устройств. Демон udevd прослушивает сообщения ядра об изменениях состояния устройств. В зависимости от конфигурационной информации, хранящейся в файле /etc/udev/udev.conf и подкаталогах, он может выполнять различные действия при обнаружении или отключении устройства. По умолчанию демон создает файлы устройств в каталоге /dev, а также пытается запускать сценарии конфигурирования сети при обнаружении новых сетевых интерфейсов.

sysfs: доступ к "сердцу" устройств

Еще одна функциональная возможность, добавленная в версии 2.6 ядра — sysfs. Это виртуальная файловая система, которая предоставляет хорошо организованную и подробную информацию обо всех доступных устройствах, их конфигурациях и состоянии. Доступ к драйверу возможен как из ядра, так и со стороны команд пользовательского уровня.

Для выяснения всей информации об устройстве (используемом им прерывании IRQ, количестве блоков поставленных в очередь записи контроллером диска и т.п.) можно исследовать каталог /sys, в котором обычно монтируется система sysfs. Один из основополагающих принципов формирования этой файловой системы состоит в том, что каждый файл в каталоге /sys должен представлять только один атрибут связанного с ним устройства. Это соглашение обеспечивает определенное структурирование в остальном хаотичного набора данных.

Первоначально информация о конфигурации устройств, если таковая существовала, хранилась в файловой системе /proc (в файле procfs, описанном в последующих подразделах этой главы). Хотя информация о процессах и ядре, связанная со временем выполнения, и в дальнейшем будет храниться в каталоге /proc, предполагается, что со временем вся информация об устройствах будет перемещена в каталог /sys.

Поскольку файловая система sysfs сравнительно нова, значительная часть ее потенциала еще не используется. Возможно, что в будущем она позволит конфигурировать устройства в реальном времени, в долговременной перспективе она, быть может, даже заменит собой всю или часть файловой системы /dev. Только время покажет, оправдаются ли эти ожидания.

Соглашения об именовании устройств

Соглашения об именовании устройств — это нечто весьма неопределенное. Во многих случаях применяется древняя методика, которую использовали еще при работе на компьютерах PDP-11 фирмы DEC.

[2] Демон udev полностью заменяет демон devfs — еще одну недавнюю попытку обеспечения таких же функциональных возможностей. Хотя в нем реализованы и не все функции демона devfs, но он почему-то считается более совершенным с архитектурной точки зрения. Как, разве вам не известно, что все эти удобные дополнительные функции devfs были всего лишь уступкой порокам?!

📖 Подробнее о последовательных портах рассказывалось в главе 27.

Имена файлов последовательных портов обычно состоят из префикса `ttyS` и числа, обозначающего конкретный интерфейс, к которому подключен порт. Иногда для терминала имеется несколько файлов устройств. Дополнительные файлы предназначены для поддержки альтернативных методов управления потоками и протоколов блокировки.

Имена накопителей на магнитной ленте часто включают не только ссылку на сам накопитель, но и указание на то, перематывается ли лента после закрытия устройства.

Файлы жестких дисков IDE называются `/dev/hd`*`XN`*, где *`X`* — буква, идентифицирующая устройство (`a` — главный диск первого IDE-интерфейса, `b` — подчиненный диск этого интерфейса, `c` — главный диск второго IDE-интерфейса и т.д.), а *`N`* — номер раздела (нумерация начинается с единицы). Например, первый раздел первого IDE-диска обычно называется `/dev/hda1`. Файлы жестких дисков SCSI называются аналогичным образом, но вместо префикса `hd` используется префикс `sd`. Для обоих типов устройств номер раздела можно опускать — это позволит получить доступ ко всему диску (например, `/dev/hda`).

SCSI-дисководам CD-ROM соответствуют файлы `/dev/scd`*`N`*, где *`N`* — идентификатор конкретного дисковода, если их несколько. Современные IDE-дисководы CD-ROM обозначаются так же, как и жесткие диски IDE (например, `/dev/hdc`).

28.3. Зачем и как конфигурировать ядро

При инсталляции системы создается базовая конфигурация, которая рассчитана на запуск практически любых приложений практически в любой аппаратной среде. В эту конфигурацию входит множество драйверов устройств и дополнительных пакетов, а значения модифицируемых параметров соответствуют режиму “для общих целей”. Тщательно настроив эту конфигурацию, можно добиться повышения производительности, защищенности и даже надежности системы.

Современные ядра Linux лучше справляются с выгрузкой ненужных драйверов, нежели их предшественники, однако большинство встроенных опций компиляции по умолчанию включено. В наши дни заниматься переконфигурированием ядра для повышения эффективности его работы уже не так актуально, как раньше, но время от времени такая потребность все же возникает.

📖 Инструкции по добавлению нового драйвера приведены в разделе 28.6.

Иногда необходимо переконфигурировать ядро, чтобы добавить поддержку нового типа устройств (т.е. нового драйвера). Нельзя просто записать файл драйвера в каталог ядра: нужно интегрировать код драйвера в структуры данных и таблицы ядра. В одних системах для этого следует добавить в конфигурационные файлы ядра сведения о новом устройстве, т.е. перестроить ядро заново. В других — достаточно выполнить специальную программу, которая сама внесет необходимые изменения.

Конфигурировать ядро не так уж сложно. Сложнее восстановить его работу, если что-то было сделано не так.

Существует четыре основных метода конфигурирования ядра Linux:

- модификация настраиваемых (динамических) параметров ядра;
- повторное создание ядра (это означает компиляцию исходных файлов ядра с возможными модификациями и дополнениями);
- динамическая загрузка новых драйверов и модулей в существующее ядро;

- передача ядру директив на этапе начальной загрузки, через загрузчик LILO или GRUB (об этих программах рассказывалось в разделе 2.3).

Каждый из перечисленных методов имеет свою область применения. Проще всего настраивать параметры ядра. Именно так обычно и поступают. Компиляция ядра — наиболее сложный метод, и к нему прибегают реже всего. Но любой метод вполне можно освоить, это лишь вопрос практики.

28.4. Конфигурирование параметров ядра Linux

Многие модули и драйверы ядра разрабатывались с учетом того, что все в системе подвержено изменению. Для повышения гибкости были созданы специальные информационные каналы, которые позволяют системному администратору динамически корректировать различные параметры, определяющие размеры внутренних таблиц ядра и его поведение в конкретных ситуациях. Все эти каналы расположены в файловой системе `/proc` (также называемой `procfs`), которая представляет собой интерфейс между ядром и приложениями пользовательского уровня. Часто большие приложения пользовательского уровня (особенно такие приложения "инфраструктуры" как базы данных) требуют определенной настройки.

Специальные файлы в каталоге `/proc/sys` позволяют просматривать и динамически изменять значения параметров ядра. Они напоминают стандартные файлы Linux, но на самом деле являются "шлюзами" к ядру. Если в одном из файлов есть значение, которое требуется изменить, можно попытаться осуществить запись в этот файл. К сожалению не все файлы доступны для записи (отображаемые права доступа не дают полной картины), а объем имеющейся документации оставляет желать лучшего. Информацию о некоторых значениях и их назначении можно получить в подкаталоге `Documentation/sysctl` каталога с исходным кодом ядра.

Например, чтобы изменить максимальное число файлов, которые одновременно могут быть открыты в системе, введите такую команду:

```
# echo 32768 > /proc/sys/fs/file-max
```

Со временем вырабатывается привычка к столь нетрадиционному способу взаимодействия с ядром, тем более что он очень удобен для изменения конфигурационных параметров. Но сразу предупредим: изменения не сохраняются при перезагрузке. В табл. 28.1 перечислены некоторые обычно настраиваемые параметры.

Более надежный способ модификации тех же параметров в большинстве систем реализован в виде команды `sysctl`. Она либо берет значения переменных из командной строки, либо читает список пар *переменная=значение* из файла. По умолчанию файл `/etc/sysctl.conf`, в котором заданы начальные (пользовательские) значения параметров, считывается на этапе начальной загрузки.

Например, команда

```
# sysctl net.ipv4.ip_forward=0
```

отключает перенаправление IP-пакетов. Обратите внимание на то, что символы косой черты в имени файла, хранящегося в структуре каталога `/proc/sys`, заменяются точками.

Таблица 28.1. Файлы каталога /proc/sys, содержащие некоторые настраиваемые параметры ядра

Каталог[a]	Файл/параметр	По умолчанию	Назначение и комментарий
C	autoeject	0	Нужно ли автоматически выталкивать компакт-диск при размонтировании дисковода CD-ROM. Подходящее средство, чтобы разбудить ночную смену операторов.
F	file-max	4096	Задает максимальное число открытых файлов. В системе, которой приходится работать с большим количеством файлов, можно попытаться увеличить это значение до 16384.
F	inode-max	16384	Задает максимальное число открытых индексных дескрипторов в одном процессе. Этот параметр может быть полезен при создании приложения, которое открывает десятки тысяч дескрипторов файлов.
K	ctrl-alt-del	0	Нужно ли перезагружаться при нажатии <Ctrl+Alt+Del>. Применение этого параметра может диктоваться личными предпочтениями или соображениями безопасности физически не защищенных серверных консолей.
K	printk ratelimit	5	Минимальный интервал между сообщениями ядра, в секундах.
K	printk_ratelimit_burst	10	Количество последовательных сообщений, которые должны быть получены, прежде чем значение минимального интервала между сообщениями printk действительно вступит в действие.
K	shmmax	32M	Задает максимальный размер совместно используемой памяти. Этот параметр требует настройки при наличии приложений, которые нуждаются в значительном объеме совместно используемой памяти.
N	conf/default/rp_filter	0	Включает проверку маршрута к исходному файлу. Этот механизм противодействия имитации соединений вынуждает ядро не обращать внимания на пакеты, полученные по “невозможным” маршрутам.
N	icmp_echo_ignore_all	0	Задает игнорирование ICMP-запросов, если равен 1. Удобен, чтобы сделать систему недоступной для ICMP-запросов.
N	icmp_echo_ignore_ broadcasts	0	Задает игнорирование широковещательных ICMP-запросов, если равен 1. Практически во всех ситуациях целесообразно устанавливать это значение равным 1.

Окончание табл. 28.1

Каталог[a]	Файл/параметр	По умолчанию	Назначение и комментарий
N	icmp_ignore_bogus_error_responses	0	Задает игнорирование ошибок неправильного форматирования сообщений ICMP. Практически во всех ситуациях целесообразно устанавливать это значение равным 1.
N	ip_forward	0	Разрешает перенаправление IP-пакетов, если равен 1. Установка этого значения равным 0 способствует повышению уровня безопасности. Если же компьютер Linux используется в качестве маршрутизатора, это значение необходимо устанавливать равным 1.
N	ip_local_port_range	32768 61000	Определяет диапазон локальных портов, выделяемый во время конфигурирования соединения. Чтобы повысить производительность серверов, которые инициируют много исходящих соединений, этот диапазон следует расширить до 1024–65000.
N	tcp_fin_timeout	60	Определяет интервал ожидания (в секундах) заключительного пакета FIN. Чтобы повысить производительность серверов, через которые проходит большой объем трафика, следует устанавливать более низкое значение (порядка 20).
N	tcp_syncookies	0	Защищает от атак волнового распространения пакетов SYN. Этот параметр необходимо включать при наличии подозрения о наличии атак отказа от обслуживания (DOS).

[a] F — /proc/sys/fs, N — /proc/sys/net/ipv4, K — /proc/sys/kernel, C — /proc/sys/dev/cdrom.

28.5. Сборка ядра Linux

Операционная система Linux развивается столь быстрыми темпами, что рано или поздно ее ядро приходится компилировать заново. Постоянно появляются новые "заплаты" к ядру, драйверы устройств и новые функции. Все это вызывает противоречивые чувства. С одной стороны, удобно иметь под рукой всегда "самое новое и свежее", а с другой — стремление идти в ногу со временем требует постоянных усилий и значительных затрат времени. Но достаточно один раз успешно скомпилировать ядро, чтобы почувствовать себя докой в этом деле.

Кроме того, если вы используете "стабильную" версию, сборка собственного ядра вряд ли потребуется. Первоначально в Linux была принята схема нумерации ядер, согласно которой вторая часть номера версии указывает на то, является ядро стабильным (четный номер) или находится в стадии разработки (нечетный номер). Например, ядро версии 2.6.6 — стабильное, в отличие от ядра версии 2.7.4. В настоящее время эта схема нумерации соблюдается не всегда. Поэтому за официальной информацией по этому вопросу лучше обратиться на сайт kernel.org. Этот сайт является также наилучшим источником исходного кода ядра Linux на тот случай, если конкретный дистрибутив (или его поставщик) не вызывают доверия.

Не чините то, что еще не поломано

Сегодня, когда новые версии ядра Linux появляются каждые несколько месяцев, а новые драйверы и "заплаты" выпускаются чуть ли не каждый день, легко потерять голову от постоянных обновлений системы. Кто из нас не испытывал это сладострастное чувство собственного превосходства, сообщая пользователям о том, что появилась новая "заплата" для ядра и ее нужно срочно инсталлировать, поэтому почтовый сервер будет выключен после обеда? Некоторые администраторы именно так доказывают свое право на высокую зарплату. В конце концов, каждому хочется хоть иногда почувствовать себя суперменом.

Хороший системный администратор тщательно взвешивает все за и против, планируя обновление ядра. Конечно, самая последняя версия — это всегда модно, но является ли она столь же стабильной, что и текущая версия? Нельзя ли подождать до конца месяца, когда будет инсталлироваться целая группа "заплат"? Важно, чтобы желание быть "на гребне волны" не шло вразрез с интересами пользователей.

Возьмите за правило устанавливать обновления и заплаты, только когда планируемая выгода (обычно определяемая на основе критериев надежности и производительности) окупит усилия и время, затрачиваемые на инсталляцию. Если выгоду оценить затруднительно, это верный признак того, что с обновлением можно подождать.

Конфигурирование параметров ядра

Процедура конфигурирования ядра Linux совершенствуется уже довольно долгое время, но она все еще достаточно примитивна в сравнении с другими операционными системами. Традиционно исходный код ядра Linux хранился в каталоге `/usr/src/linux`, а для построения ядра требовались права суперпользователя. Компиляция ядер версий 2.4, 2.6 и последующих может выполняться обычными пользователями, и поэтому в настоящее время правильнее хранить исходный код в специализированном каталоге, таком как `/usr/local/src/kernel` или *~имя_пользователя*`/kernel`.

В данной главе мы используем строку *путь_к_исходным_текстам_ядра* в качестве шаблона любого каталога, который выбран для хранения исходного кода ядра. В любом случае, чтобы можно было создавать ядро системы, должен быть инсталлирован пакет его исходных кодов. Об инсталляции пакетов рассказывается в разделе 11.1.

Весь процесс сосредоточен вокруг файла `.config`, находящегося в каталоге исходных кодов ядра. В данном файле приведена вся информация, касающаяся конфигурирования ядра, но его формат очень труден для понимания. Воспользуйтесь пособием по параметрам конфигурирования ядра, приведенном в файле

путь_к_исходным_текстам_ядра`/Documentation/Configure.help`

для выяснения смысла различных параметров.

Чтобы не редактировать файл `.config` напрямую, в Linux предусмотрено несколько целевых модулей утилиты `make`, реализующих различные интерфейсы конфигурирования ядра. Если в системе инсталлирована оболочка KDE, то удобнее всего воспользоваться командой `make xconfig`. Аналогично, если вы используете оболочку GNOME, вероятно лучше всего использовать команду `make gconfig`. Эти команды отображают окно конфигурации, в котором можно выбрать драйверы устройств, добавляемые к ядру (или компилируемые в качестве загружаемых модулей).

В отсутствие графической среды на помощь придет команда `make menuconfig`, работающая на основе библиотеки `curses`[3]. Наконец, есть старая команда `make config`, которая выводит запрос на изменение каждого конфигурационного параметра и не позволяет изменять ранее установленные значения. Мы рекомендуем пользоваться командой `make xconfig` или `make gconfig`, если используемая среда их поддерживает, и командой `make menuconfig` в противном случае. Применения команды `make config` лучше избегать.

При переносе конфигурации существующего ядра в новую версию (или файловую систему) ядра можно использовать команду `make oldconfig` для считывания ранее использовавшегося файла `config` и определения только тех параметров, которые изменились или являются новыми.

Все эти команды довольно просты, так как их действие сводится к установке нужных опций. Тем не менее, они мало подходят для ситуации, когда нужно поддерживать несколько версий ядра для различных платформ или аппаратных конфигураций.

В любом случае генерируется файл `.config`, имеющий примерно такой вид:

```
# Automatically generated make config: don't edit
# Code maturity level options

CONFIG_EXPERIMENTAL=y

#
# Processor type and features #
# CONFIG_M386 is not set
# CONFIG_M486 is not set
# CONFIG_M586 is not set
# CONFIG_M586TSC is not set
CONFIG_M686=y
CONFIG_X86_WP_WORKS_OK=y
CONFIG_X86_INVLPG=y
CONFIG_X86_BSWAP=y
CONFIG_X86_POPAD_OK=y
CONFIG_X86_TSC=y
CONFIG_X86_GOOD_APIC=y
...
```

Как видите, содержимое файла не слишком понятно. Не дается никаких пояснений относительно того, что означают параметры `CONFIG_*`. По сути, каждая строка `CONFIG_*` указывает на активизацию того или иного параметра ядра. Значение `y` говорит о том, что соответствующий компонент компилируется вместе с ядром; значение `m` означает, что компонент станет загружаемым модулем.

Не все компоненты можно сконфигурировать в виде модулей. Сведений об этом в файле `.config` нет, поэтому нужно самостоятельно покопаться в документации. Не так-то легко узнать и назначение параметров `CONFIG_*`.

Компиляция ядра

Формирование файла `.config` — самый важный этап конфигурирования ядра Linux, но нужно выполнить еще целый ряд действий, прежде чем будет получено готовое ядро.

[3] Библиотека `curses` содержит функции, формирующие псевдографический интерфейс в терминальном окне.

Схема всего процесса такова:

- перейти в каталог верхнего уровня, содержащий исходные коды ядра;
- выполнить команду `make xconfig`, `make gconfig` или `make menuconfig`;
- выполнить команду `make dep` (не требуется для ядер версии 2.6.x и более поздних);
- выполнить команду `make clean`;
- выполнить команду `make`;
- выполнить команду `make modules_install`;
- скопировать файл `arch/i386/boot/bzImage` под именем `/boot/vmlinuz`;
- скопировать файл `arch/i386/boot/System.map` под именем `/boot/System.map`;
- отредактировать файл `/etc/lilo.conf` (LILO) или `/boot/grub/grub.conf` (GRUB) и добавить в него конфигурационные параметры для нового ядра;
- если используется программа LILO, выполнить команду `/sbin/lilo` для инсталляции обновленного загрузчика.

Выполнять команду `make clean` не всегда нужно, но удаление всего лишнего позволит избежать множества проблем.

28.6. Добавление драйвера устройства в Linux

В Linux драйверы устройств распространяются в одной из трех форм:

- "заплаты" к конкретной версии ядра;
- загружаемого модуля;
- инсталляционного сценария или пакета, устанавливающего соответствующие "заплаты".

Наиболее распространенная форма — инсталляционный сценарий или пакет. Если вам повезло, и вы располагаете таким сценарием или пакетом для нового устройства, достаточно выполнить инструкции по выполнению сценария или установке пакета, чтобы ядро распознало новое устройство.

В большинстве случае установку "заплаты" для конкретной версии ядра можно выполнить с помощью следующей команды[4]:

```
# cd путь_к_исходным_текстам_ядра ; patch -p1 < файл_заплаты
```

Внесение изменений в различные более ранние версии ядра может оказаться неудачным, но драйвер должен работать.

Если это не так, скорее всего, придется вручную интегрировать драйвер нового устройства в дерево каталогов исходного кода ядра. Наиболее вероятная причина возникновения подобной ситуации — сборка собственного драйвера ядра, а не происки коммерческого поставщика устройства, который не позаботился о предоставлении соответствующих инсталляционных сценариев. Ниже мы рассмотрим, как вручную добавить к ядру драйвер некоего вымышленного сетевого устройства "snarf". В действительности

[4] Естественно, чтобы можно было модифицировать дерево каталогов ядра, должен быть инсталлирован пакет его исходных кодов.

в Linux этот процесс достаточно утомителен, особенно по сравнению с некоторыми другими версиями UNIX.

В подкаталоге drivers дерева файлов исходного кода ядра можно найти подкаталог, соответствующий типу добавляемого устройства. Вот как выглядит список существующих подкаталогов:

```
$ ls -F путь_к_исходным_текстам_ядра/drivers
acorn/     char/     i2c/        Makefile  net/      s390/    telephony/
acpi/      dio/      ide/        md/       nubus/    sbus/    usb/
atm/       fc4/      ieee1394/   media/    parport/  scsi/    video/
block/     gsc/      input/      message/  pci/      sgi/     zorro/
bluetooth/ hil/      isdn/       misc/     pcmcia/   sound/
cdrom/     hotplug/  macintosh/  mtd/      pnp/      tc/
```

Чаще всего драйверы добавляются в каталоги block, char, net, scsi, sound и usb. В них хранятся драйверы блочных устройств (например, IDE-дисков), символьных устройств (например, последовательных портов), сетевых устройств, SCSI-плат, звуковых плат и USB-устройств, соответственно. В других каталогах находятся, в частности, драйверы для самих шин (pci, nubus и zorro); маловероятно, чтобы потребовалось добавлять в них файлы. Некоторые каталоги содержат платформо-зависимые драйверы (macintosh, s390, acorn). Есть каталоги для специализированных драйверов (atm, isdn, telephony).

Поскольку наше устройство "snarf" является сетевым, мы поместим его драйвер в каталог drivers/net. Потребуется модифицировать следующие файлы:

- drivers/net/Makefile, чтобы драйвер мог быть скомпилирован;
- drivers/net/Kconfig, чтобы имя устройства появилось в списке конфигурируемых параметров.

После помещения файлов с расширениями .c и .h в каталог drivers/net/snarf нужно добавить запись о драйвере в файл drivers/net/Makefile. Вот эта строка (она располагаются ближе к концу файла):

```
obj-$(CONFIG_SNARF_DEV) += snarf/
```

Эта строка добавляет драйвер (хранящийся в каталоге snarf/) в процесс компиляции ядра.

Когда запись добавлена в файл Makefile, нужно убедиться в том, что устройство будет доступно для конфигурирования при настройке ядра. Все сетевые устройства должны быть перечислены в файле drivers/net/Kconfig. В нашем случае требуется записать в него приведенную ниже строку, чтобы драйвер был добавлен либо как загружаемый модуль, либо как часть ядра (в соответствии с тем, что было заявлено в файле Makefile):

```
config SNARF_DEV tristate 'Snarf device support'
```

Первый компонент, следующий за ключевым словом config, — имя конфигурационного макроса, которое должно совпадать с компонентом, следующим за ключевым словом CONFIG в файле Makefile. Ключевое слово tristate означает, что драйвер может стать загружаемым модулем. Если это не соответствует действительности, вместо tristate необходимо указать ключевое слово bool. Следующим элементом инструкции является строка, которая будет отображаться в окне конфигурации. Это может быть произвольный текст, но он должен идентифицировать конфигурируемое устройство.

Распознание устройства ядром

Как при компиляции в ядро нового драйвера устройства указать ядру, что оно должно использовать новый драйвер? В версиях ядра, предшествовавших версии 2.6, эта задача была весьма трудоемкой и требовала знаний в области программирования. Недавние архитектурные изменения, внесенные в модель драйверов устройств, позволяют драйверам стандартным образом связывать себя с ядром.

Подробное описание этого процесса выходит за рамки данной книги, но в результате драйверы устройств, созданные для версии 2.6 (и последующих) выполняют свою регистрацию с помощью макроса `MODULE_DEVICE_TABLE`. Этот макрос создает соответствующие скрытые связи, чтобы такие утилиты, как `modprobe` (описывается ниже), могли активизировать новые устройства в ядре.

28.7. Загружаемые модули ядра

Благодаря наличию загружаемых модулей появляется возможность подключать и отключать драйверы устройств и другие компоненты ядра непосредственно в процессе работы системы. Это значительно облегчает инсталляцию драйверов, поскольку исполняемый код ядра не нужно менять. Кроме того, уменьшается размер ядра, так как ненужные драйверы просто не загружаются.

Поддержка загружаемых модулей реализуется путем вставки в ядро одной или нескольких документированных "точек подключения", которые доступны драйверам устройств. Существует команда пользовательского уровня `insmod`, которая дает ядру указание загрузить новые модули в память. Команда `rmmod` выгружает драйверы.

Загружаемые драйверы очень удобны, но они не обеспечивают стопроцентную безопасность. При каждой загрузке и выгрузке модуля существует риск нарушить работоспособность ядра[5]. Во избежание нарушения работы системы не рекомендуется загружать и выгружать непроверенные модули.

В Linux почти все программные компоненты можно сделать загружаемыми модулями. Исключение составляет драйвер устройства, на котором расположена корневая файловая система, а также драйвер мыши PS/2.

Загружаемые модули обычно хранятся в каталоге `/lib/modules/`*версия*, где последняя часть имени — это версия ядра Linux, сообщаемая командой `uname -r`. Получить список загруженных в данный момент модулей можно с помощью команды `lsmod`:

```
# lsmod
Module          Size     Used by
ppp             21452    0
slhc            4236     0 [ppp]
ds              6344     1
i82365          26648    1
pcmcia_core     37024    0 [ds    i82365]
```

Как видно из списка, в системе установлены модули контроллера PCMCIA, драйвер PPP и модули сжатия PPP-заголовков.

Приведем пример загрузки вручную модуля ядра `snarf`, который был установлен в предыдущем разделе:

```
# insmod /путь/snarf.ko
```

[5] В действительности риск очень незначителен. Он может быть настолько мал, что часто им можно просто пренебречь.

Загружаемым модулям ядра можно также передавать конфигурационные параметры, например:

```
# insmod /путь/snarf.ko io=0xXXX irq=X
```

После того как модуль вручную добавлен к ядру, удалить его можно только по явному запросу или при перезагрузке системы. Для удаления нашего модуля подойдет команда `rmmod snarf`.

Команду `rmmod` можно использовать в любой момент времени, но она будет работать, только если число действующих ссылок на модуль (это число указано в столбце `Used by` (Используется) результата выполнения команды `lsmod`) равно 0.

Модули ядра Linux могут загружаться полуавтоматически с помощью команды `modprobe`, которая является оболочкой команды `insmod` и распознает межмодульные зависимости, параметры модулей, процедуры инсталляции и выгрузки. Эта команда просматривает файл `/etc/modprobe.conf`, чтобы узнать правила обработки каждого модуля. Можно динамически создать файл `/etc/modprobe.conf`, соответствующий текущей конфигурации ядра, выполнив команду `modprobe -c`. Эта команда генерирует длинный файл, который выглядит примерно так:

```
#This file was generated by: modprobe -c
path[pcmcia]=/lib/modules/preferred
path[pcmcia]=/lib/modules/default
path[pcmcia]=/lib/modules/2.6.6
path[misc]=/lib/modules/2.6.6
...
# Aliases
alias block-major-1 rd
alias block-major-2 floppy
...
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
...
alias dos msdos
alias plip0 plip
alias ppp0 ppp
options ne io=x0340 irq=9
```

Инструкции `path` указывают на то, где находится конкретный модуль. Можно модифицировать или добавлять записи данного типа, если нужно хранить модули в нестандартных каталогах.

Инструкция `alias` обеспечивает привязку старших номеров блочных устройств, старших номеров символьных устройств, файловых систем, сетевых устройств и сетевых протоколов к соответствующим модулям.

Строки с ключевым словом `options` не генерируются динамически. Они задают параметры, передаваемые модулю при загрузке. Например, в следующей строке модулю устройства "snarf" сообщается его адрес ввода-вывода и вектор прерываний[6]:

```
options snarf io=0xXXX irq=X
```

[6] Если система установлена на нестандартном персональном компьютере, то довольно сложно подобрать такую конфигурацию, в которой векторы прерываний и адреса ввода-вывода не перекрываются. Чтобы узнать текущую конфигурацию системы, нужно просмотреть содержимое файлов `/proc/interrupts` и `/proc/ioports`. Как правило, для современного оборудования ПК всех основных типов перекрытие векторов прерываний и адресов ввода-вывода не создает особых проблем.

Команда `modprobe` понимает также инструкции `install` и `remove`. С их помощью указываются команды, выполняемые, когда соответствующий модуль подключается к работающему ядру или отключается от него.

28.8. "Горячее" подключение

В то время как загружаемые модули решают задачу динамического добавления и удаления драйверов устройств из ядра, функция "горячего" подключения Linux экспортирует информацию о доступности устройства на уровень пространства пользователя. Эта функция позволяет процессам пользователя реагировать на такие события, как подключение оснащенных USB-интерфейсом цифровых камер или миникомпьютеров. Например, может требоваться автоматически выполнить копирование изображений на локальный диск или, не выключая ПК, синхронизировать календарь миникомпьютера. Идея "горячего" подключения устройств заключается в том, что при этом от пользователя не должно требоваться выполнение никаких дополнительных действий, кроме физического подключения устройства.

Начиная с ядра версии 2.6, "горячее" подключение поддерживается шинами и драйверами, которые разработаны с учетом использования `sysfs`. "Горячее" подключение и рассмотренные ранее файловая система `sysfs` и регистрация драйверов устройств тесно связаны между собой.

В нынешней реализации функции "горячего" подключения при каждом обнаружении добавления или устройства ядро выполняет процесс пользователя, указанный параметром `/proc/sys/kernel/hotplug` (обычно `/sbin/hotplug`). `/sbin/hotplug` — сценарий интерпретатора, который для выполнения действий в соответствии с событием вызывает ориентированного на тип устройства агента из каталога `/etc/hotplug/`. Например, при добавлении сетевого интерфейса был бы выполнен сценарий `/etc/hotplug/net.agent` для перевода интерфейса в активное состояние. Сценарии в каталоге `/etc/hotplug` можно добавлять или редактировать для настройки поведения системы при "горячем" подключении устройств.

По соображениям безопасности или по каким-либо другим причинам может требоваться, чтобы система не предпринимала никаких действий при "горячем" подключении. В подобных случаях для предотвращения выполнения действий в ответ на соответствующие события устройства можно добавить в файл `/etc/hotplug/blacklist`.

И наоборот, можно принудительно вызвать выполнение действий "горячего" подключения, создавая файлы "handmap", подобные `/etc/hotplug/`*`тип`*`.handmap`. (Если создаете новый файл устройства, убедитесь, что *тип* не конфликтует с существующими именами.)

28.9. Настройка параметров начальной загрузки

После того как рабочее ядро собрано, возможно, понадобится передать ему на этапе начальной загрузки специальные конфигурационные параметры, например имя корневого устройства или инструкцию, указывающую на необходимость опроса нескольких Ethernet-плат. Функция передачи этих параметров ядру возложена на загрузчик (LILO или GRUB).

Статические параметры конфигурации, которые должны использоваться при каждой начальной загрузке системы, задаются в файле `/etc/lilo.conf` или `/boot/grub/grub.conf`,

в зависимости от используемого загрузчика. Подробная информация об этом была приведена в разделе 2.3.

Если отредактировать файл конфигурации загрузчика не представляется возможным (что-то было сделано не так, и компьютер не загружается), нужные параметры придется указать вручную. Например, в строке приглашения LILO можно ввести:

```
LILO: linux root=/dev/hda1 ether=0,0,eth0 ether=0,0,eth1
```

Эта строка заставляет программу LILO загрузить ядро, помеченное как "linux", использовать в качестве корневого устройство `/dev/hda1` и проверить наличие двух Ethernet-плат.

В случае загрузчика GRUB команды выглядят так:

```
grub> kernel /vmlinuz root=/dev/hda1 ether=0,0,eth0 ether=0,0,eth1
grub> boot
```

Еще одна распространенная ситуация, в которой целесообразно использовать параметры начальной загрузки — проверка логических номеров устройств (logical unit number, LUN) в сети хранения данных (SAN). По умолчанию ядро Linux осуществляет зондирование только с LUN равным 0, чего недостаточно, если в данной среде сети хранения данных представлены различными LUN. (Для уточнения этой информации обратитесь к своему администратору SAN или поставщику оборудования.) В этой ситуации ядру необходимо указать, сколько логических номеров LUN необходимо зондировать после зондирования, выполненного во время начальной загрузки.

Например, чтобы ядро 2.4.x выполняло зондирование первых 8 номеров LUN, строка загрузки должна выглядеть следующим образом:

```
grub> kernel /vmlinuz root=/dev/hda1 max_scsi_luns=8
grub> boot
```

В ядрах версий 2.6.x и последующих имя параметра изменилось:

```
grub> kernel /vmlinuz root=/dev/hda1 max_luns=8
grub> boot
```

28.10. Рекомендуемая литература

- Bovet, Daniel P., and Marco Cesati. *Understanding the Linux Kernel (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2006.
- Corbet, Jonathan, et al. *Linux Device Drivers (3rd Edition)*. Sebastopol, CA: O'Reilly Media, 2005. Эта книга доступна также по адресу `lwn.net/Kernel/LDD3`.
- Love, Robert. *Linux Kernel Development (2nd Edition)*. Indianapolis, IN: Novell Press, 2005. (Роберт Лав, *Разработка ядра Linux, 2-е издание* (пер. с англ., ИД "Вильямс", 2006 г.))

28.11. Упражнения

28.1. Опишите, что делает ядро? Объясните разницу между загрузкой драйвера в качестве модуля и его статической компиляцией в ядро.

☆ 28.2. Проверьте значения нескольких параметров, перечисленных в табл. 28.1, путем обращения к элементам файловой системы `/proc` и вызова команды `sysctl`. Измените два параметра одним способом и прочтите их новые значения дру-

гим способом. Убедитесь, что поведение системы действительно изменилось. (Необходим доступ с правами суперпользователя.)

☆ 28.3. Вы по дешевке купили Ethernet-плату для портативного компьютера, которая позволяет подключаться к сети через параллельный порт. Какие действия необходимо предпринять, чтобы операционная система Linux распознала новую плату? Нужно ли встроить поддержку платы непосредственно в ядро или ее следует реализовать в виде модуля? Почему? (Вопрос на засыпку: если ваша почасовая ставка составляет $80, во сколько обойдется установка этой "дешевой" платы?)

☆ 28.4. Только что была выпущена новая версия ядра Linux, и вы хотите обновить все компьютеры в местной лаборатории (общим числом около 50, не все из них идентичны). Какие моменты следует учесть? В какой последовательности выполнять действия? Какие проблемы могут возникнуть и как их решить?

☆☆ 28.5. Настройте ядро с помощью команды `make xconfig` или `make menuconfig` и создайте исполняемый образ ядра. Инсталлируйте ядро и перезагрузите систему. Включите выполнение команды `dmesg` на этапе инициализации системы и сравните результаты, выдаваемые старым и новым ядрами. (Необходим доступ с правами суперпользователя.)

Глава 29

Демоны

Демон — это фоновый процесс, который выполняет определенную функцию или системную задачу. В соответствии с господствующим в UNIX и Linux принципом модульности демоны являются программами, а не компонентами ядра. Многие демоны запускаются на этапе начальной загрузки системы и продолжают работать все время, пока система включена. Остальные демоны запускаются по мере необходимости и работают столько, сколько от них требуется.

Слово “демон” (daemon) как компьютерный термин впервые употребил англичанин Мик Бейли (Mick Bailey), который в начале 60-х гг. работал программистом в Массачусетсском технологическом институте[1]. Для объяснения смысла и правильного написания слова “daemon” Мик цитировал Оксфордский словарь английского языка. Согласно его объяснениям, слова “daemon” и “demon” имеют общий корень, но первое — более древнее, сохранившее первоначальное значение. Слово “daemon” дословно означает “гений, дух-хранитель человека, формирующий его характер”. В этом смысле демоны являются не воплощениями зла или добра, а независимыми существами со своими собственными намерениями и желаниями. Из системы CTSS, в работе над которой Мик Бейли принимал участие, этот термин перекочевал в Multics, а затем в UNIX и Linux, где демоны стали настолько популярными, что для управления ими понадобился “супердемон” (`xinetd` или `inetd`).

В этой главе вкратце рассмотрены наиболее распространенные демоны Linux. Не все они поставляются с дистрибутивами Linux, и не каждый демон, входящий в состав того или иного дистрибутива, здесь упомянут. Знакомство с основными демонами поможет начинающим администраторам не только лучше разобраться в особенностях работы

[1] Эту историю рассказал нам Деннис Ритчи со слов Джерри Зальтцера (Jerry Saltzer) из МТИ.

Linux, но и не бояться каверзных вопросов пользователей вроде: “Для чего нужен демон klogd?”.

До того как был создан демон inetd, все демоны запускались на этапе начальной загрузки и работали непрерывно (точнее, блокировались в ожидании заданий). Со временем в систему вводились все новые и новые демоны. Их появилось столько, что начали возникать проблемы с производительностью. В ответ специалисты университета Беркли разработали inetd — “супердемон”, запускающий другие демоны по мере необходимости. Демон inetd сделал модель супердемонов настолько популярной, что она до сих остается наиболее распространенным способом сведения к минимуму количества выполняющихся на сервере процессов. В большинстве версий UNIX и Linux теперь постоянно выполняющиеся демоны используются только в сочетании с inetd.

Системные администраторы должны быть хорошо знакомы с целым рядом демонов: либо потому, что они требуют много внимания со стороны администратора, либо потому, что играют значительную роль в повседневной работе системы. Некоторым демонам, которые здесь описаны одной-двумя строчками, посвящены целые разделы в других главах этой книги. Там, где необходимо, мы даем соответствующие ссылки.

В начале этой главы мы познакомим читателей с двумя очень важными системными демонами (init и cron), а затем перейдем к рассмотрению демонов xinetd и inetd. Далее вкратце будет описано большинство демонов, с которыми системному администратору, скорее всего, придется столкнуться при работе в четырех наших демонстрационных дистрибутивах.

29.1. Демон init: первичный процесс

Демон init — это первый процесс, который запускается после загрузки ядра, и во многих отношениях это самый важный демон. Он всегда имеет идентификатор процесса 1 и является предком всех пользовательских и почти всех системных процессов.

В ходе начальной загрузки демон init либо переключает систему в однопользовательский режим, либо начинает выполнять сценарии, переводящие систему в многопользовательский режим. Если система загружается в однопользовательском режиме, демон init вызывает сценарии запуска после того, как однопользовательский интерпретатор команд завершает свою работу по команде exit или при нажатии клавиш <Ctrl+D>.

В многопользовательском режиме демон init отвечает за то, чтобы с каждым устройством, через которое возможна регистрация в системе, был связан соответствующий процесс. Запросы на регистрацию, поступающие через последовательные порты, обычно обрабатываются одной из разновидностей программы getty (agetty, mgetty, mingetty; рассматриваются в конце раздела 27.9). Демон init также контролирует процедуру графической регистрации, которая позволяет пользователям входить непосредственно в систему X Window.

Помимо задач, связанных с регистрацией, в обязанности демона init также входит удаление процессов-зомби, которые в противном случае обязательно бы скапливались в системе. Его роль в этом процессе описывалась в разделе 4.3.

Демон init поддерживает различные *уровни выполнения*, которые определяют, какой набор системных ресурсов следует задействовать. Существуют семь уровней, пронумерованных от 0 до 6. Обозначение “s” является синонимом уровня 1 (однопользовательский режим). Характеристики всех уровней содержатся в файле /etc/inittab.

📖 О файле inittab рассказывалось в конце раздела 27.9.

Свой начальный уровень выполнения демон init обычно считывает из файла /etc/initab, но этот уровень также может передаваться ему и в виде аргумента из системного загрузчика. Если задается уровень "s", init переходит в однопользовательский режим. В противном случае он находит в файле /etc/inittab записи, соответствующие указанному уровню, и выполняет содержащиеся в них команды.

Команда telinit позволяет менять уровень выполнения во время работы системы. Например, команда telinit 4 заставит демон init перейти на уровень 4 (который в рассматриваемых нами системах не используется). Самым полезным аргументом команды telinit является q, который вынуждает init осуществлять повторное считывание файла /etc/inittab.

В дистрибутивах Linux обычно реализуется дополнительный уровень абстракции, позволяющий отдельным программным пакетам инсталлировать собственные сценарии запуска, не модифицируя системный файл inittab. Перевод демона init на определенный уровень вызывает выполнение соответствующих сценариев с аргументами start или stop. Более подробное описание этого механизма приводилось в начале раздела 2.5.

29.2. Демоны cron и atd: планирование запуска команд

Демон cron (в Red Hat называется crond) отвечает за выполнение команд в заданное время. Он обрабатывает файлы с расписанием задач (crontab-файлы), создаваемые как пользователями, так и администраторами.

Демон cron часто применяют в административных целях, например для управления журнальными файлами и ежедневной чистки файловой системы. Этот демон настолько важен для системных администраторов, что мы посвятили ему целую главу 8.

Демон atd выполняет задания, запланированные командой at. В состав большинства версий Linux также входит планировщик anacron, который выполняет задания не в указанное время, а через определенные промежутки времени. Он особенно полезен на тех системах, которые не всегда включены, таких как ноутбуки.

29.3. Демоны xinetd и inetd: управление демонами

xinetd и inetd — это демоны, которые управляют другими демонами. Они запускают своих демонов-клиентов только тогда, когда для них есть работа, и позволяют им корректно завершаться сразу же, как только те выполнят все свои задачи.

Традиционный демон inetd пришел к нам из мира UNIX, но в большинстве дистрибутивов Linux он был заменен демоном xinetd, созданным Паносом Цироготисом (Panos Tsirogotis) и включающим средства защиты, которые ранее были доступны только путем использования tcpd (так называемого пакета "TCP-упаковщиков"). xinetd также обладает более надежными средствами защиты от атак типа "отказа в обслуживании" (Denial Of Service), более удобными возможностями управления журналами и более гибким языком настройки.

К сожалению, конфигурационный файл inetd не совместим с аналогичным файлом xinetd. Сначала мы расскажем о более часто встречающемся демоне xinetd, а затем уже — в отдельном разделе — о демоне inetd.

Среди рассматриваемых нами дистрибутивов стандартный демон inetd используется только в Debian и Ubuntu; в дистрибутивах RHEL, Fedora и SUSE по умолчанию применяется демон xinetd. Конечно, стандартные настройки в любой системе можно поменять, но нет никаких причин делать это.

xinetd и inetd работают только с теми демонами, которые предоставляют службы по сети. Чтобы перехватывать запросы к своим клиентам, демоны xinetd и inetd подключаются к сетевым портам, которые при обычных обстоятельствах обслуживались бы статическими демонами. При появлении соединения они запускают соответствующий демон и подключают его стандартные каналы ввода-вывода к сетевому порту. Этот момент должен обязательно учитываться при написании совместимых демонов.

Работа некоторых демонов (например, тех, что связаны с NIS и NFS) зависит от механизма RPC, который был разработан и реализован компанией Sun в качестве средства совместного использования данных в гетерогенной сетевой среде. Назначениями портов для RPC-демонов управляет демон portmap, рассматриваемый далее в главе.

Некоторые демоны могут выполняться как традиционным образом (т.е. запускаться однократно и работать до выключения системы), так и под управлением демона xinetd/inetd. Все описываемые в этой главе демоны, которые совместимы с xinetd/inetd, помечены значком .

Демон xinetd/inetd отвечает за функционирование множества распространенных сетевых служб, поэтому он играет важную роль в обеспечении безопасности системы. Следует убедиться в том, что активизированы только действительно необходимые и проверенные службы. В новой системе почти наверняка придется изменять стандартную конфигурацию, чтобы отключить ненужные или нежелательные службы.

Конфигурирование демона xinetd

Главным конфигурационным файлом демона xinetd по традиции считается файл /etc/xinetd.conf, хотя в состав дистрибутивов обычно также входит и конфигурационный каталог /etc/xinetd.d. Отдельные пакеты могут помещать свои конфигурационные файлы в этот каталог, не волнуясь о перекрытии конфигураций других пакетов.

Ниже показан пример установки параметров по умолчанию и конфигурирования службы FTP на системе Red Hat Enterprise:

```
defaults
{
   instances = 60
   log_type = SYSLOG authpriv
   log_on_success = HOST PID
   log_on_failure = HOST
   cps = 25 30
}
service ftp
{
   # Число экземпляров не ограничено, поскольку демон wu.ftpd
   # самостоятельно управляет загрузкой.
   socket_type = stream
   protocol = tcp
   wait = no
   user = root
   server = /usr/sbin/wu.ftpd
   server_args = -a
   instances = UNLIMITED
```

```
    only_from = 128.138.0.0/16
    log_on_success += DURATION
}
includedir /etc/xinetd.d
...
```

В табл. 29.1 приводится краткий перечень основных параметров демона `xinetd`.

Таблица 29.1. Основные конфигурационные параметры `xinetd`

Параметр	Значение	Описание
`bind`	*IP-адрес/хост*	Интерфейс, через который должна быть доступна данная служба
`cps`	*Время ожидания в виде числа*	Позволяет ограничивать общее количество соединений в секунду
`disable`	`yes`/`no`	Позволяет отключать службу; это проще, чем заключать все описание в комментарии
`include`	*файл*	Считывает указанный файл как дополнительный конфигурационный файл
`includedir`	*каталог*	Считывает все файлы, содержащиеся в указанном каталоге
`instances`	*число*/`UNLIMITED`	Максимальное число функционирующих одновременно экземпляров данной службы
`log_on_failure`	*специальное значение*[а]	Информация, которая должна регистрироваться в журнале событий в случае сбоя или отказа в доступе (т.е. в случае неудачной попытки подключения)[б]
`log_on_success`	*специальное значение*[а]	Информация, которая должна регистрироваться в журнале событий в случае успешной попытки подключения[б]
`log_type`	*специальное значение*[а]	Позволяет задавать журнальный файл или конфигурировать параметры Syslog
`max_load`	*число*	Отключает службу, если показатель средней загруженности превышает указанный порог
`nice`	*число*	Значение `nice` порождаемых серверных процессов
`no_access`	*список адресов*	Отключает службу для указанных IP-адресов
`only_from`	*список адресов*	Принимает запросы только с указанных адресов
`per_source`	*число*	Предельное число экземпляров службы для каждого удаленного компьютера
`protocol`	`tcp/udp`	Коммуникационный протокол службы
`server`	*файл*	Бинарный файл сервера
`server_args`	*строка*	Аргументы командной строки для сервера[в]
`socket_type`	`stream/dgram`	Тип используемых сокетов: `stream` для TCP-служб, `dgram` для UDP-служб
`user`	*имя пользователя*	Пользователь, от имени которого должен запускаться демон
`wait`	`yes/no`	Должна ли `xinetd` ждать, пока завершится серверный демон

[а] Поддерживаемых значений слишком много для того, чтобы их стоило воспроизводить в этой таблице.

[б] Обратите внимание на то, что директива `USERID`, используемая в этих параметрах, вынуждает `xinetd` выполнять запросы `IDENT` для всех соединений, что часто приводит к значительным задержкам.

[в] В отличие от `inetd`, `xinetd` не требует, чтобы первым аргументом была команда сервера.

Некоторые параметры демона xinetd допускают операции присваивания вида += и -= (как в случае параметра log_on_success FTP-сервера в показанном выше примере). Это позволяет редактировать стандартное значение параметра, а не менять его полностью. Вообще говоря, такой синтаксис разрешен везде, где это имеет смысл. Для каждой службы необходимо лишь несколько параметров. Они соответствуют тем информационным элементам, которые содержатся в файле inetd.conf.

Списки адресов в параметрах only_from и no_access могут задаваться в нескольких форматах. Наиболее удобны IP-адреса в CIDR-нотации с явной маской (как в данном примере) и доменные адреса вида boulder.colorado.edu или .colorado.edu (обратите внимание на начальную точку). Группы спецификаций отделяются друг от друга пробелами.

Демон xinetd может регистрировать сообщения либо прямо в файле, либо в системе Syslog. Поскольку в случае интенсивно эксплуатируемого сервера объем журнальной информации может оказаться достаточно большим, из соображений производительности имеет смысл прибегнуть к первому варианту. Но помните, что регистрация сообщений в файле менее безопасна, чем регистрация на удаленном сервере через систему Syslog, так как хакер, проникший в локальную систему, может подделать журнальные файлы.

Демон xinetd располагает рядом необычных средств. Например, он может предоставлять услуги по переадресации запросов на внутренний хост, недоступный внешнему миру. Подробнее об этом можно прочитать на man-странице демона.

Конфигурирование демона inetd

Debian и Ubuntu являются единственными среди основных дистрибутивов Linux, в которых до сих пор используется традиционный демон inetd. Для того чтобы определить, какие сетевые порты следует контролировать, демон inetd считывает файл /etc/inetd.conf. В этом конфигурационном файле содержится во многом почти такая же информация, как и в файле xinetd.conf, но используется табличный (а не "атрибут/значение") формат. Ниже показан пример такого файла в системе Debian.

```
# Пример файла /etc/inetd.conf - из системы Debian

#:INTERNAL: внутренние службы
#echo    stream   tcp   nowait   root  internal
#echo    dgram    udp   wait     root  internal
...
#time    stream   tcp   nowait   root  internal
#time    dgram    udp   wait     root  internal

#:STANDARD: это стандартные службы
#:BSD: Shell, login, exec and talk are BSD protocols. (Оболочка, механизм
регистрации, exec и talk - все это протоколы BSD)

#:MAIL: служба почты, служба новостей и служба uucp
imap2    stream   tcp   nowait   root  /usr/sbin/tcpd /usr/sbin/imapd
imaps    stream   tcp   nowait   root  /usr/sbin/tcpd /usr/sbin/imapd

#:INFO: информационные службы
ident    stream   tcp   wait  identd   /usr/sbin/identd identd
...
```

```
#:OTHER: другие службы
swat        stream   tcp      nowait.400  root     /usr/sbin/swat swat
# finger    stream   tcp      nowait      nobody   /usr/sbin/tcpd in.fingerd -w
391002/1-2  stream   rpc/tcp  wait        root     /usr/sbin/famd fam
...
```

В первой колонке содержатся имена служб. Демон `inetd` отображает имена служб на номера портов при помощи либо файла `/etc/services` (для UDP- и TCP-служб), либо файла `/etc/rpc` и демона `portmap` (для RPC-служб). RPC-службы можно отличить по названию, которое у них имеет формат *имя/номер*, и наличию значения `rpc` в третьей колонке. Например, в последней строке показанного выше фрагмента файла содержится информация об RPC-службе.

Обычно единственной другой RPC-службой, которой управляет демон `inetd`, является служба `mountd`, представляющая собой демон монтирования NFS. В дистрибутивах Linux этот демон, похоже, в основном запускается по-старому (т.е. на этапе начальной загрузки), так что в файле `inetd.conf` вполне может и вообще не быть никаких RPC-служб.

На компьютере с несколькими сетевыми интерфейсами можно предварять название службы списком разделенных запятыми IP-адресов или доменных имен, обозначающих интерфейсы, через которые демон `inetd` должен ожидать поступления запросов к службе. Например, строка

```
inura:time    stream  tcp  nowait    root   internal
```

разрешает работу службы времени только через интерфейс, с которым связано имя `inura` в DNS, NIS или в файле `/etc/hosts`.

Во второй колонке определяется тип сокета, с которым будет работать демон: `stream` или `dgram`. Сокеты типа `stream` используются в TCP-службах (ориентированных на установление соединений), а сокеты типа `dgram` — в UDP-службах; однако, в некоторых службах, таких как `bind`, используются сокеты и того, и другого типа.

В третьей колонке указывается используемый службой коммуникационный протокол. Список допустимых протоколов содержится в файле `protocols` (который обычно расположен в каталоге `/etc`). Здесь почти всегда указывается протокол `tcp` или `udp`. В случае RPC-служб перед протоколом еще также указывается тип службы, например `rpc/tcp`.

Если описываемая служба может одновременно обрабатывать несколько запросов (а не завершаться после обработки одного запроса), в четвертой колонке должно стоять ключевое слово `wait`. Оно позволяет порожденному демону захватывать контроль над портом на время своего выполнения. Демон `inetd` будет дожидаться завершения такого демона, чтобы вновь начать контролировать порт. Ключевое слово `nowait` заставляет демон `inetd` работать в непрерывном режиме и порождать новый экземпляр подчиненного демона при поступлении очередного запроса. Выбор одного из двух режимов должен соответствовать фактическому поведению демона и не может делаться произвольно.

Запись `nowait.400` в описании утилиты `swat` означает, что демон `inetd` должен порождать как минимум 400 экземпляров подчиненного демона в минуту. Установка по умолчанию более консервативна: 40 экземпляров в минуту. Учитывая природу этой службы (Web-администрирование пакета Samba), не совсем понятно, зачем было увеличено пороговое значение.

В пятой колонке указывается пользователь, от имени которого должен выполняться демон. С точки зрения безопасности демон лучше запускать от имени пользователя, отличного от пользователя `root`, если это, конечно, возможно. В показанном выше фраг-

менте, например, демон in.fingerd выполнялся бы от лица пользователя nobody (если бы отвечающая за него строка не была закомментирована).

В остальных полях указываются полный путь к демону и аргументы его командной строки. Ключевое слово internal обозначает службу, реализация которой предоставляется самим демоном inetd.

Демоны многих служб в приведенном примере запускаются не напрямую, а через tcpd. Программа tcpd регистрирует все попытки установить соединение и осуществляет контроль доступа на основании их источника. В принципе, защищаться при помощи tcpd должны все службы. Показанная в данном примере конфигурация является потенциально небезопасной, потому что утилита swat, предназначенная для настройки совместного доступа к файлам, не защищена[2].

В стандартном файле inetd.conf, поставляемом с Debian, серверы для rlogin, telnet, finger и rexec уже даже больше не перечисляются. Более подробную информацию по связанным с безопасностью вопросам можно найти в разделе 20.9.

Отредактировав файл /etc/inetd.conf, нужно отправить демону inetd сигнал HUP для того, чтобы он выполнил повторное считывание своего конфигурационного файлы и чтобы все внесенные изменения вступили в силу. После этого следует немного подождать и потом проверить журнальные файлы на наличие сообщений об ошибках, произошедших в результате изменений (демон inetd регистрирует ошибки в Syslog, в разделе daemon). И, конечно же, напоследок нужно протестировать все добавленные новые службы, дабы удостовериться в том, что они работают правильно.

📖 О Syslog более подробно рассказывалось в главе 10.

Файл services

После добавления новой службы в файл inetd.conf или xinetd.conf может также понадобиться внести запись для нее и в файл /etc/services. Этот файл используется несколькими стандартными библиотечными алгоритмами, которые отображают имена служб на номера портов и наоборот.

```
tcpmux   1/tcp              # мультиплексор TCP-портов
echo     7/tcp
echo     7/udp
...
ssh      22/tcp             #Протокол удаленной регистрации SSH
ssh      22/udp             #Протокол удаленной регистрации SSH
smtp     25/tcp   mail
rlp      39/udp   resource  # размещение ресурсов
name     42/tcp             # спецификация IEN 116
domain   53/tcp             # сервер домена имен
domain   53/udp
...
```

Формат каждой строки выглядит так:

имя порт/протокол псевдонимы # комментарий

[2] Если программа tcpd не используется, первым аргументом командной строки демона всегда должно быть имя самого демона. Это не особенность демона inetd, но традиционное для UNIX соглашение, поддерживаемое интерпретаторами команд.

Службы обычно перечисляются в порядке следования номеров их портов, хотя такой порядок не является обязательным. В поле *имя* указывается символическое имя службы (то, которое было указано в файле `inetd.conf` или `xinetd.conf`). В поле *порт* указывается номер порта, который обычно прослушивает данная служба. Если службой управляет демон `inetd`, тогда здесь указывается порт, который будет прослушивать демон `inetd`[3].

В поле *протокол* указывается протокол, используемый службой (на практике это всегда протокол `tcp` или `udp`). Если служба поддерживает и протокол TCP, и протокол UDP, в файл должна включаться отдельная строка для каждого из них (как в случае службы `time` в показанном выше примере). В поле *псевдонимы* содержатся дополнительные имена службы (например, службу `whois` также можно отыскать и по имени `nicname`).

Демон `portmap`: отображение номеров RPC-служб на номера портов TCP и UDP

Демон `portmap` отображает номера RPC-служб на номера портов TCP/IP, которые прослушивают их серверы. Когда RPC-сервер запускается, он регистрирует себя при помощи демона `portmap` и перечисляет службы, которые он поддерживает, и порт, через который с ним можно связаться. Клиенты запрашивают демон `portmap` и узнают, как можно получить доступ к нужному серверу.

Такая система позволяет устанавливать соответствие между номерами портов и названиями служб. Это еще один уровень абстракции, более высокий, чем в файле `services`, но вместе с ним появляются дополнительные сложности (и проблемы безопасности), тогда как реальных проблем он не решает.

Если демон `portmap` останавливается, все зависящие от него службы (включая `inetd` и NFS) нужно перезапускать снова. На практике это означает, что пришло время перезагрузить систему. Для того чтобы демон `inetd` обрабатывал RPC-запросы правильно, демон `portmap` должен запускаться раньше него.

29.4. Демоны ядра

Из концептуальных соображений некоторые компоненты ядра Linux представлены так, будто они являются пользовательскими процессами. В ядрах более старых версий эти процессы могут иметь низкое значение PID и имена, начинающиеся на “k”, например `kupdate`, `kswapd`, `keventd` или `kapm`. В ядрах версии ниже 2.6 правила именования согласуются хуже, но `ps` всегда отображает имена потоков ядра в квадратных скобках.

По большей части эти процессы отвечают за различные аспекты ввода-вывода, управление памятью и синхронизацию дисковых кэш-буферов. Системному администратору нельзя вмешиваться в их работу и следует оставлять их в покое[4].

[3] Номера портов назначаются не в произвольном порядке. Все компьютеры должны согласовать друг с другом, какие службы за какими портами закреплены. В противном случае запросы будут постоянно идти не на те порты. Конфигурируя службу специально для своей системы, назначьте ей большой номер порта (больше чем 1023), которого еще нет в файле `services`.

[4] Если системный администратор знаком с реализацией ядра, он иногда может изменять приоритеты выполнения этих процессов. Однако стандартной процедурой администрирования эта операция никак не является.

В табл. 29.2 приводится краткий перечень функций всех являющихся основными на текущий момент демонов. Демоны, в именах которых присутствует параметр *N* (когда их отображают при помощи `ps`), в системах с множеством ЦП выполняются отдельно на каждом ЦП; параметр *N* показывает, какая копия выполняется именно на данном ЦП.

Таблица 29.2. Основные демоны ядра (для ядер версии 2.6)

Демон	Функция
`ksoftirqd/`*N*	Обрабатывает программные прерывания при высокой нагрузке
`kacpid`	Обслуживает подсистему ACPI
`kblockd/`*N*	Блокирует работу подсистем
`aio/`*N*	Повторяет асинхронные запросы ввода-вывода
`kswapd`*N*	Переносит страницы в область подкачки
`ata/`*N*	Выполняет обработку для поддержки последовательного интерфейса ATA
`scsi_eh_`*N*	Выполняет обработку ошибок SCSI
`kjournald`	Обеспечивает журнализацию файловых систем
`events/`*N*	Выполняет обычную обработку рабочих очередей событий

Еще одним системным демоном в этой категории, хотя и имеющим нестандартное имя, является демон `mdrecoveryd`. Он представляет собой часть подсистемы управления RAID-массивами.

`klogd`: чтение сообщений ядра

Демон `klogd` отвечает за считывание журнальных сообщений из буфера сообщений ядра и их пересылку в Syslog, так чтобы они могли попасть в свой конечный пункт назначения. Он также может обрабатывать сообщения самостоятельно, если его сконфигурировать соответствующим образом. Подробнее об этом рассказывалось в конце раздела 10.2.

29.5. Демоны печати

Сегодня широко используется целых несколько систем печати, и каждая из них имеет свой собственный набор предназначенных для печати команд и демонов. Иногда эти наборы объединяются, а иногда — функционируют на одной единственной системе в виде множества вариантов.

Демон `cupsd`: планировщик для системы CUPS

Система CUPS (Common UNIX Printing System — Общая система печати в UNIX) предоставляет переносимое средство печати путем реализации протокола IPP (Internet Printing Protocol — Протокол печати через Internet) версии 1.1. Она позволяет удаленным пользователям выполнять печать в своих офисах (или наоборот) посредством Web-интерфейса. CUPS стала довольно популярной и потому сегодня используется в качестве стандартного диспетчера печати в большинстве систем. Она является достаточно гибкой для того, чтобы допускать удаленную аутентификацию.

📖 О CUPS более подробно рассказывалось в главе 23.

Демон lpd: управление процессом печати

Демон lpd отвечает за работу уже устаревшей BSD-подсистемы спулинга печати. Он принимает задания от пользователей и порождает процессы, которые фактически выполняют печать. Кроме того, демон lpd также отвечает за пересылку заданий на печать с и на удаленные системы. Он иногда может зависать и в таком случае его нужно перезапускать вручную.

В системе может присутствовать либо исходная версия демона lpd, либо его улучшенная версия из пакета LPRng. Подробнее об этих альтернативах рассказывалось в главе 23.

29.6. Демоны обслуживания файлов

Перечисленные ниже демоны являются компонентами либо сетевой файловой системы NFS, либо системы совместного использования файлов Samba. Здесь приводится лишь краткое описание их функций, потому что об NFS подробно рассказывалось в главе 16, а о Samba — в главе 26.

rpc.nfsd: обслуживание файлов

Демон rpc.nfsd выполняется на файловых серверах и отвечает за обработку запросов, поступающих от клиентов NFS. В большинстве реализаций NFS демон nfsd является частью ядра, "замаскированной" под процесс ради простоты планирования. В Linux имеются две реализации демона, одна из которых следует указанному соглашению, а вторая работает как пользовательский процесс. Реализация на уровне ядра более популярна и по умолчанию используется в большинстве дистрибутивов.

Демон rpc.nfsd принимает один единственный аргумент, который указывает, какое количество копий самого себя он должен порождать. Правильно выбрать это значение довольно сложно. О том, как именно это можно сделать рассказывалось в конце раздела 16.2.

rpc.mountd: обработка запросов на монтирование

Демон rpc.mountd принимает от потенциальных клиентов NFS запросы на монтирование файловых систем и проверяет, имеет ли клиент право монтировать указанные каталоги. Для этого демон mountd обращается к файлу /var/state/nfs/xtab.

amd и automount: монтирование файловых систем по запросу

Демоны amd и automount — это утилиты автоматического монтирования, которые, прежде чем смонтировать файловую систему NFS, ждут, когда процесс обратится к ней. Впоследствии, если к файловой системе не было обращения в течение определенного промежутка времени, демоны демонтируют ее.

Демоны автоматического монтирования очень удобны в крупных организациях, когда десятки и сотни файловых систем совместно используются по сети. Эти демоны повышают стабильность сети и уменьшают сложность ее конфигурирования, так как все компьютеры в сети могут использовать одинаковые конфигурационные файлы amd или

automountd. Об использовании стандартного демона монтирования в Linux подробно рассказывалось в разделе 16.6.

rpc.lockd и rpc.statd: управление блокировками NFS

Демоны rpc.lockd и rpc.statd — это два разных демона, но они всегда работают в паре. Первый отвечает за блокировку файлов NFS; второй позволяет процессам контролировать состояние компьютеров, на которых работает NFS. Демон statd используется демоном lockd для принятия решения о том, когда следует установить связь с удаленным компьютером.

rpciod: кэширование блоков данных NFS

Демон rpciod отвечает за кэширование запросов чтения и записи на клиентах NFS. Он выполняет буферизацию как в режиме опережающего чтения, так и в режиме фоновой записи, что значительно повышает производительность NFS. Этот демон аналогичен демонам biod и nfsiod, которые встречаются в других версиях UNIX, но реализован иначе.

rpc.rquotad: дистанционное управление квотами

Демон rpc.rquotad позволяет удаленным пользователям проверять свои квоты в файловых системах, смонтированных через NFS. Реально квоты реализуются на сервере. Демон rpc.rquotad просто заставляет команду quota работать корректно.

smbd: файловый сервер и сервер печати для клиентов Windows

Демон smbd — это поставляемый в пакете Samba файловый сервер и сервер печати. Он обеспечивает совместный доступ к файлам и принтерам по протоколу SMB/CIFS. Подробнее об этом рассказывалось в разделе 26.6.

nmbd: сервер имен NetBIOS

Демон nmbd — еще один компонент пакета Samba. Он отвечает на запросы к службе имен NetBIOS, генерируемые Windows-компьютерами. Он также реализует протокол просмотра, который Windows-компьютеры используют для заполнения папки My Network Places (Моя сеть), и делает там видимыми сетевые диски, совместно используемые на локальном хосте. Демон nmbd может также использоваться в качестве сервера WINS.

29.7. Демоны административных СУБД

Несколько демонов связано с административной СУБД NIS, которая описывалась в главе 17. Эта СУБД была создана компанией Sun, но нашла широкое применение в системах других поставщиков, включая Linux.

ypbind: поиск серверов NIS

Демон ypbind работает на всех клиентах и серверах NIS. Он устанавливает местонахождение сервера NIS, на который могут направляться запросы. Сам демон не обрабатывает запросы, а просто сообщает клиентским программам, к какому серверу следует обращаться.

ypserv: сервер NIS

Демон ypserv работает на всех серверах NIS. Он принимает запросы от клиентов и отвечает на них. О конфигурировании систем, в которых работает демон ypserv, рассказывалось в разделе 17.4.

rpc.ypxfrd: пересылка баз данных NIS

Демон rpc.ypxfrd пересылает базы данных NIS на подчиненные серверы очень эффективным образом. Подчиненный сервер инициирует пересылку командой ypxfr. Сразу после изменения базы данных на главном сервере она немедленно рассылается на все подчиненные серверы.

lwresd: упрощенный сервер библиотечного распознавателя

Демон lwresd представляет собой быстрый механизм кэширования поисков адреса по имени хоста и имени хоста по адресу. С ним контактирует соответствующий распознаватель, который является частью стандартных библиотек системы и вызывается напрямую многими программам. Библиотека и демон взаимодействуют посредством простого протокола UDP.

nscd: демон кэширования для службы имен

Демон nscd кэширует результаты вызовов функций стандартной библиотеки языка C, в частности функций семейств getpw*, getgr* и gethost*, которые ищут данные, традиционно хранившиеся в файлах passwd, group и hosts. В наши дни круг потенциальных источников информации шире и включает такие службы, как NIS и DNS. Демону nscd не известно, откуда поступают данные; он просто кэширует результаты запросов, используя их для ускорения последующих вызовов функций. Правила кэширования задаются в файле /etc/nscd.conf.

29.8. Демоны электронной почты

Помимо основных систем доставки почты, таких как sendmail и Postfix, обе из которых используются повсеместно, существует несколько демонов, которые упрощают удаленный доступ к почтовым ящикам.

sendmail: транспортировка электронной почты

В число задач программы sendmail входит прием сообщений от пользователей и удаленных компьютеров, подстановка адресов, раскрытие псевдонимов и пересылка

почты в сети Internet. Это очень важный и сложный демон. Подробнее он рассматривался в главе 18.

smtpd: демон протокола SMTP

Демон `smtpd` прослушивает порт 25, принимая через него все входящие электронные сообщения и пересылая их в соответствующую серверную систему транспортировки для дальнейшей обработки. Об использовании `smtpd` в системах `sendmail` и Postfix подробно рассказывалось в разделах 18.3 и 18.16.

popd: базовый сервер почтовых ящиков

Демон `pop` реализует протокол POP (Post Office Protocol — протокол почтового офиса). Этот протокол в основном используется в не-Linux (т.е., Windows) системах для получения электронной почты.

imapd: высококлассный сервер почтовых ящиков

Демон `imapd` реализует протокол IMAP (Internet Message Access Protocol — протокол доступа к Internet-сообщениям), который является более мощной и более функциональной альтернативой протокола POP. Этот демон позволяет пользователям персональных компьютеров (или пользователям Linux, которые работают с поддерживающими IMAP почтовыми клиентами) получать доступ к своим хранящимся на Linux-сервере почтовым ящикам из самых разных мест. Дополнительную информацию о протоколе IMAP можно найти на Web-сайте `www.imap.org`.

29.9. Демоны удаленной регистрации и удаленного выполнения команд

Возможность регистрироваться и выполнять команды удаленно через сеть была одной из первых причин разработки сетевой технологи в UNIX и по сей день остается важным компонентом системного администрирования. К сожалению, сообществу пользователей UNIX понадобилось несколько десятков лет на то, чтобы зрело оценить проблемы безопасности этой технологии. Сегодня в производственных системах должен использоваться протокол SSH (`sshd`) и больше практически ничего.

sshd: безопасный сервер удаленной регистрации

Демон `sshd` функционирует подобно демону `in.rlogind`, но все данные (в том числе аутентификационные) передаются в зашифрованном виде. Поддерживается множество алгоритмов шифрования. В связи с тем, что современная сеть Internet весьма небезопасна, доступ в систему через Internet следует разрешать только этому демону и ни в коем случае демонам `in.rlogind` и `in.telnetd`. О демоне `sshd` более подробно рассказывалось в разделе 20.11.

in.rlogind: устаревший сервер удаленной регистрации

Демон in.rlogind некогда являлся стандартным демоном удаленной регистрации. Когда демон in.rlogind вызывается из демона inetd, он пытается автоматически аутентифицировать удаленного пользователя путем изучения содержимого его локального пользовательского файла ~/.rhosts и системного файла /etc/hosts.equiv. Если ему это удается (т.е. если автоматическая аутентификация проходит успешно), пользователь регистрируется в системе напрямую. В противном случае демон in.rlogind запускает программу login, которая запрашивает у пользователя пароль. Из-за этой упрощенной схемы аутентификации демон in.rlogind представляет серьезную угрозу безопасности. Подробнее об этом речь шла в разделе 20.9.

in.telnetd: еще один сервер удаленной регистрации

Демон in.telnetd похож на демон in.rlogind, но только он использует протокол TELNET. Этот протокол позволяет двум компьютерам (клиенту и серверу) согласовывать порядок обмена данными и взаимодействовать в дуплексном режиме. Для низкоскоростных и малонадежных каналов это более предпочтительный выбор, чем демон in.rlogind. Программа telnet, как и rlogin, передает пароли в незашифрованном виде по сети, поэтому использовать ее не рекомендуется. Тем не менее, она все еще широко поддерживается, и не только в Linux.

in.rshd: сервер удаленного выполнения команд

Демон in.rshd обрабатывает запросы на удаленное выполнение команд, поступающие от программ rsh и rcmd. Процесс аутентификации, инициируемый этим демоном, напоминает действия демона in.rlogind, за одним исключением: если автоматическая аутентификация не срабатывает, запрос отклоняется без предоставления пользователю возможности указать пароль. Демон in.rshd является также сервером для программы удаленного копирования rcp. В последнее время считается, что демон in.rshd, как и демон in.rlogind, представляет угрозу безопасности, и потому его все время отключают. Подробнее об этом рассказывалось в разделе 20.9.

29.10. Демоны начальной загрузки и конфигурирования

В 80-е гг. прошлого столетия мир UNIX охватила мания внедрения бездисковых рабочих станций. Эти компьютеры загружали операционную систему по сети и выполняли все дисковые операции ввода-вывода посредством удаленных файловых систем, таких как NFS. После стремительного падения цен на жесткие диски и повышения их быстродействия интерес к бездисковым рабочим станциям быстро угас. О бездисковой эпохе сейчас напоминает только плеяда демонов, предназначенных для поддержки бездисковых компьютеров, и причудливая организация файловых систем большинства производителей.

Для любопытных: о самих бездисковых системах более подробно рассказывалось в разделе 11.2.

Несмотря на то что бездисковые рабочие станции практически не встречаются сегодня, протоколы бездисковой загрузки используются другими устройствами. Большинство сетевых концентраторов и сетевых принтеров загружается с помощью описанных ниже демонов.

dhcpd: динамическое назначение адресов

Протокол DHCP (Dynamic Host Configuration Protocol — протокол динамического конфигурирования хостов) дает возможность персональным компьютерам, портативным и другим мобильным системам получать информацию о своих IP-адресах, стандартных шлюзах и серверах имен на этапе начальной загрузки. `dhcpd` — это демон, который реализует этот протокол в Linux. Подробнее о протоколе DHCP рассказывалось в разделе 12.9. Протокол PXE (Pre-boot eXecution Environment — предзагрузочная среда выполнения), который представляет собой более сложную реализацию протокола DHCP, позволяет совместимым компьютерам получать загрузочную информацию по сети, не используя локальные загрузочные устройства (см. раздел 11.1).

in.tftpd: простейший сервер пересылки файлов

Демон `in.tftpd` реализует протокол пересылки файлов, похожий на тот, с которым работает демон `ftpd`, но гораздо более простой. Многие бездисковые станции пользуются протоколом TFTP для загрузки системного ядра с сервера. Демон `in.tftpd` не выполняет аутентификацию, но обычно его функции ограничены обслуживанием файлов в одном каталоге (как правило, `/tftpboot`). Поскольку данные, помещаемые в этот каталог, доступны для всей сети, в нем должны находиться только загрузочные файлы, и он должен быть закрыт для записи.

rpc.bootparamd: усовершенствованная поддержка бездисковых систем

Используя файл `/etc/bootparams`, демон `rpc.bootparamd` сообщает бездисковым клиентам о том, где они должны искать свои файловые системы. Услугами этого демона часто пользуются компьютеры, которые получают свои IP-адреса по протоколу RARP и монтируют файловые системы через NFS.

hald: демон уровня аппаратных абстракций

Демон `hald` собирает информацию о системном оборудовании из нескольких источников и предоставляет динамический список устройств через D-BUS.

udevd: сериализация уведомлений о подключении устройств

Демон `udevd` представляет собой небольшой компонент динамической системы именования устройств `udev`. Он отвечает за надлежащую сериализацию событий “горячего” подключения устройств, которые ядро иногда может передавать на рабочую область пользователя в неупорядоченном виде.

29.11. Другие сетевые демоны

Все описываемые ниже демоны для обработки запросов используют протоколы Internet. Однако, многие из этих "Internet-демонов" на самом деле большую часть своего времени тратят все-таки на обслуживание локальных запросов.

talkd: сервер программы talk

Демон `talkd` обрабатывает запросы на подключение, поступающие от программы `talk`. Получив запрос, он связывается с другим компьютером для того, чтобы наладить сетевое соединение между двумя пользователями, которые запустили программу `talk`.

snmpd: сервер дистанционного управления сетями

Демон `snmpd` отвечает на запросы, посылаемые по протоколу SNMP (Simple Network Management Protocol — простой протокол управления сетью). Этот протокол был разработан с целью стандартизации распространенных операций управления сетями. О нем более подробно рассказывалось в разделе 19.8.

ftpd: сервер пересылки файлов

Демон `ftpd` обрабатывает запросы, поступающие от программы `ftp`, которая позволяет пересылать файлы через Internet. Во многих организациях этот демон отключают, обычно — из соображений безопасности. Демон `ftpd` можно настроить так, чтобы все пользователи имели право пересылать файлы в систему и копировать файлы из нее.

📖 Подробнее о демоне `ftpd` рассказывалось в разделе 21.6.

Для систем Linux доступно несколько разных реализаций демона `ftpd`. Тем, кто планирует запускать высокопроизводительный сервер или нуждается в усовершенствованных функциональных возможностях, таких как управление нагрузкой, стоит поискать альтернативные версии стандартного демона `ftpd`. Одной из наиболее популярных альтернативных версий является программа WU-FTPD, разработанная в Вашингтонском университете. Подробнее о ней можно узнать на сайте `www.wu-ftpd.org`.

rsyncd: синхронизация файлов между группой компьютеров

Демон `rsyncd` в действительности представляет собой просто ссылку на команду `rsync`; опция `--daemon` превращает ее в серверный процесс. Демон `rsyncd` упрощает синхронизацию файлов, находящихся на разных компьютерах. По сути, он представляет собой более эффективную и надежную в плане безопасности версию программы `rcp`. Утилита `rsync` является настоящей находкой для системных администраторов. В книге она описана в нескольких местах. Общая информация об утилите и советы, касающиеся совместного использования системных файлов, приводились в конце раздела 17.3. Во многих системах эта утилита используется также для инсталляции программного обеспечения.

routed: ведение таблиц маршрутизации

Демон routed управляет таблицами маршрутизации, которые используются протоколами TCP/IP для передачи пакетов по сети. Демон routed выполняет только динамическую маршрутизацию; статические маршруты (задаются командой route) никогда не изменяются при помощи routed. Демон routed является довольно простым и неэффективным, поэтому мы рекомендуем пользоваться им только в особых случаях. Подробнее о нем рассказывалось в разделе 13.4.

gated: ведение сложных таблиц маршрутизации

Демон gated понимает несколько протоколов маршрутизации, включая RIP, который используется демоном routed. Демон gated передает информацию о маршрутах между разными протоколами и отличается большой гибкостью. Кроме того, он гораздо меньше загружает сеть, чем демон routed. Подробнее о демоне gated рассказывалось в разделе 13.5.

named: сервер DNS

Демон named — это самый популярный сервер службы DNS (Domain Name System — служба доменных имен). Он отображает имена компьютеров на сетевые адреса и выполняет много других функций, пользуясь распределенной базой данных, которую ведут демоны named. Подробная информация об этом демоне приводилась в главе 15.

syslogd: обработка журнальных сообщений

Демон syslogd играет роль координирующего центра для журнальных сообщений, выдаваемых системными программами и демонами. До появления демона syslogd другие демоны либо выдавали свои диагностические сообщения прямо на системную консоль, либо вели собственные журнальные файлы. Теперь сообщения пересылаются демону syslogd через библиотечную функцию syslog. Демон сортирует сообщения в соответствии с правилами, заданными системным администратором.

📖 Подробнее о Syslog рассказывалось в разделе 10.4.

in.fingerd: поиск пользователей

Демон in.fingerd предоставляет информацию о пользователях, зарегистрировавшихся в системе. При необходимости он может сообщить более подробные сведения об отдельных пользователях. Демон in.fingerd делает не так уж много: он просто принимает поступающие строки и передает их локальной программе finger.

Среди прочей информации о пользователе программа finger сообщает его регистрационные данные, содержимое поля GECOS в файле /etc/passwd, а также содержимое пользовательских файлов ~/.plan и ~/.project.

Если система подключена к Internet и в ней работает демон in.fingerd, то информацию, выдаваемую программой finger, может получить кто угодно. Демон in.fingerd позволяет реализовывать ряд весьма полезных служб (например, поиск по “белым страницам” Internet), но хакеры пользуются им в своих корыстных целях. Некоторые организации в целях предотвращения злоупотреблений отключают демон in.fingerd, а некоторые просто ограничивают объем выдаваемой им информации. Не следует думать, что раз демон in.fingerd является простым, значит, он является и безопасным: имен-

но этот демон в 1988 г. использовал знаменитый вирус-червь для проведения своих злонамеренных атак типа “переполнения буфера”.

httpd: Web-сервер

Демон `httpd` позволяет компьютеру становится Web-сервером. Демон `httpd` может пересылать своим клиентам текстовые, графические и звуковые данные. Об обслуживании Web-страниц более подробно рассказывалось в главе 21.

29.12. NTPD: ДЕМОН СИНХРОНИЗАЦИИ ВРЕМЕНИ

С развитием компьютерных сетей появилась и необходимость в согласовании текущего времени между подключенными к сети компьютерами. Таймеры синхронизации важны как для сопоставления записей в журнальных файлах в случае “взлома” системы защиты, так и для работы различных приложений конечных пользователей, например, в случае совместной разработки программных проектов или обработки финансовых транзакций.

Демон `ntpd`[5] реализует протокол NTP (Network Time Protocol — протокол сетевого времени), который позволяет компьютерам синхронизировать часы между собой с точностью до миллисекунд. Первая реализация протокола NTP появилась приблизительно в 80-х гг. прошлого столетия и позволяла выполнять синхронизацию часов с точностью всего лишь до нескольких сотен миллисекунд. Сегодня новая модель часов ядра способна синхронизировать время с точностью до одной наносекунды. Самая последняя версия этого протокола (версия 4, документирована в RFC2783) совместима со всеми предыдущими версиями, легко конфигурируется и имеет несколько функций защиты.

NTP-серверы организовываются в иерархию, каждый раздел в которой называется *уровнем* (stratum). На серверах первого уровня время обычно устанавливается по каким-нибудь внешним эталонным часам, таким как часы радиоприемника или обычные (атомные) часы. На серверах второго уровня время устанавливается по часам серверов первого уровня; также серверы второго уровня выступают в роли центров распределения времени. Всего предусмотрено до 16 уровней. Чтобы определить свой собственный уровень, сервер времени просто добавляет единицу (1) к значению уровня сервера, имеющего самый высокий уровень из тех, по которым он синхронизирует время. В 1999 г., согласно данным проведенного Нельсоном Минаром (Nelson Minar) опроса среди пользователей NTP-сети, на уровне 1 находилось 300 серверов, на уровне 2 — 20 000 серверов, а на уровне 3 — свыше 80 000 серверов[6].

Сегодня NTP-клиенты могут получать доступ к целому ряду различных служб эталонного времени наподобие служб WWV и GPS. По указанному ниже адресу доступен составленный Национальным институтом стандартов и технологий США (National Institute of Standards and Technology) список авторитетных серверов времени в Internet:

```
www.boulder.nist.gov/timefreq/service/time-servers.html
```

Почти все Internet-провайдеры предоставляют свои собственные службы времени, которые должны быть ближе их клиентам в масштабе сети (и точнее, если NTP работает правильно).

---

[5] Ранее этот демон был также известен под именем “xntpd”.

[6] Более подробная информация доступна на странице `www.media.mit.edu/~nelson/research/ntp-survey99`.

Демон ntpd реализует как клиентскую, так и серверную сторону протокола NTP. При запуске он считывает файл /etc/ntp.conf. В этом конфигурационном файле можно указывать права доступа, клиентские сети, службы времени, групповых клиентов, параметры общей конфигурации и параметры аутентификации; но не пугайтесь — там все понятно.

В состав дистрибутивов Debian и Ubuntu демон ntpd по умолчанию, похоже, не входит, но без труда доступен через apt-get. Все текущее программное обеспечение также можно получить по адресу ntp.isc.org.

Кроме того, еще можно пользоваться универсальной утилитой ntpdate для установки системных часов с NTP-сервера. Однако такой подход считается хуже подхода с использованием ntpd, потому что чреват превращением времени в один бесконечный поток. Особенно опасен внезапный перевод часов назад, поскольку программы иногда воспринимают время как монотонно возрастающую функцию. ntpd использует более "мягкий" системный вызов adjtimex для сглаживания последствий корректировки системных часов и предотвращения больших "скачков во времени" вперед или назад. adjtimex замедляет ход системных часов: как только системное время совпадет с целевым временем, замедление хода прекращается, и часы снова начинают работать в нормальном режиме.

29.13. Упражнения

☆ 29.1. С помощью команды ps определите, какие демоны сейчас активны на вашей системе. Также посмотрите, какие вообще демоны доступны, заглянув в файл inetd.conf или xinetd.conf. Опишите, что делает каждый демон, где он запускается, может ли одновременно запускаться несколько копий этого демона (и запущены ли они сейчас), а также все остальные имеющиеся у данного демона атрибуты.

☆ 29.2. Установите и настройте в лаборатории демон сетевого времени ntpd.

а) Как узнать, правильное ли время установлено в системе?

б) С помощью команды date вручную переведите системные часы на 15 секунд назад. Насколько скоро произойдет синхронизация времени (и произойдет ли вообще)?

в) Переведите системные часы на месяц вперед. Какова будет реакция демона ntpd?

(Необходим доступ с правами суперпользователя.)

☆☆ 29.3. В лаборатории с помощью команды netstat определите, какие из портов локального компьютера находятся в состоянии прослушивания (LISTEN).

а) Совпадает ли информация, которую вернула команда netstat, с той, что содержится в файле inetd.conf/xinetd.conf? Если есть расхождения, то почему они возникли?

б) Установите на другом компьютере утилиту nmap (которая описывалась в разделе 20.10) и проверьте с ее помощью порты своего компьютера. Можно ли таким способом узнать дополнительную информацию, не сообщаемую в явном виде командой netstat?

Глава 30

Стратегии и методики управления и администрирования

Можно иметь самую профессиональную команду администраторов в мире, но без соответствующего технического руководства будет худо всем, и пользователям в том числе. Эта глава посвящена рассмотрению нетехнических аспектов создания успешной организации по поддержке информационных технологий, а также нескольких технических вопросов (типа продумывания инфраструктуры, использования диагностических систем и т.д.), затрагивающих связанную с управлением часть системного администрирования.

Большинство представленных в этой главе тем и идей не относятся ни к какой определенной области. Они касаются одинаково как одного системного администратора, работающего на полставки, так большой группы профессионалов, работающих на полную ставку и обслуживающим какой-нибудь большой IT-проект. Они подобно овощам подходят практически к каждому "блюду". (Для очень больших организаций с сотнями IT-сотрудников в этой главе также приводится краткое описание ориентированной на процессы архитектуры управления IT-ресурсами, которая называется "Information Technology Interface Library" (Библиотека интерфейсов информационных технологий) и больше всего подходит именно для организаций большого размера. См. раздел 30.14.)

Помимо советов по управлению, мы также включили в эту главу разделы, посвященные следующим темам: документация (раздел 30.8), диагностические системы (раздел

30.9), восстановление после аварий (раздел 30.10), разработка стратегий (раздел 30.11), правовые вопросы (раздел 30.12), патенты на программное обеспечение (раздел 30.13), попытки стандартизации (раздел 30.14), идеология Linux (раздел 30.15) и имеющие отношение к Linux организации, конференции и семинары для системных администраторов (раздел 30.17).

30.1. Что необходимо для того, чтобы все были довольны

Системное администрирование, по сути, представляет собой сферу услуг, которыми пользуются как люди, так и компьютерами. Некоторые заинтересованные больше в технических деталях администраторы порой считают пользователей своих систем не более чем "нюансами", которые мешают настоящей работе. Это бесполезная и неправильная точка зрения; на самом деле задачей системного администратора является обеспечивать пользователей такой инфраструктурой IT-технологий, которая позволит им работать эффективно и качественно.

Всем нам доводилось поработать в службе технической поддержки в тот или иной период своей карьеры. Попробуйте сейчас поставить себя на место пользователей и ответить на следующие вопросы: "Что было не так?" и "Что бы следовало изменить?".

Пользователи довольны тогда, когда:

- их компьютеры настроены, работают нормально и позволяют им регистрироваться в системе;
- их файлы данных имеют тот вид, в котором они их оставляли;
- необходимые им приложения установлены и работают так, как и должны;
- дружелюбный и знающий свое дело сотрудник отдела поддержки доступен всегда, когда он им нужен.

Пользователи хотят, чтобы все это у них было 24 часа в сутки и желательно бесплатно. Пользователям не нравятся:

- простои, как запланированные, так и незапланированные;
- обновления, которые приводят к появлению неожиданных и несовместимых изменений;
- непонятные сообщения как от системы, так и от администраторов;
- долгие объяснения по поводу того, почему что-то не работает.

Когда что-нибудь ломается, пользователи хотят знать, когда это исправят. И все. Их не волнует, ни связано ли это с отказом жесткого диска или генератора, ни почему этот отказ мог произойти; эту информацию следует оставлять для отчетов руководству.

С точки зрения пользователя отсутствие новостей является хорошей новостью. Система либо работает, либо нет, и если она не работает, то ему без разницы почему. Наши клиенты наиболее счастливы тогда, когда они даже не знают о нашем существовании! Печально, но так оно и есть.

Пользователи являются нашими главными клиентами, но не менее важно, чтобы довольны были и сотрудники. Хороших администраторов трудно найти, и их потребности обязательно должны учитываться в организациях при разработке систем администрирования.

Системные администраторы и другие технические специалисты довольны тогда, когда:

- их компьютеры и системы поддержки настроены и работают нормально;
- у них есть ресурсы, необходимые им для выполнения их работы;
- в их распоряжении самые новейшие и самые совершенные программные и аппаратные средства;
- им не приходится постоянно отвлекаться во время работы;
- они могут применять творческий подход без излишнего вмешательства и придирок начальства;
- количество рабочих часов и уровень стресса находятся в пределах нормы.

Сотрудникам технического профиля для того, чтобы они могли продолжать работать, чека в конце месяца не достаточно. Им необходимо чувствовать, что у них есть определенная свобода в том, что они делают, и что их ценят коллеги, начальство и пользователи.

Требования для довольных клиентов и довольных сотрудников IT-отдела в кое-чем похожи. Но в кое-чем они совершенно не совпадают и даже являются напрямую противоположными. Начальник должен обязательно проверять, все ли из таких отличных ожиданий можно сделать совместимыми, и достижимы ли они вообще.

В некоторых организациях сам начальник общается с клиентами, чтобы технический персонал не отвлекался и мог выполнять свою "настоящую" работу. Согласно нашему опыту, такая схема, как правило, является неудачным решением. Она не позволяет техническому персоналу напрямую знакомиться с потребностями клиентов, что часто заканчивается тем, что начальник дает обещания, которые технические специалисты просто не в силах выполнить.

30.2. Компоненты функциональной IT-организации

С увеличением организации, занимающейся предоставлением услуг в сфере системного администрирования, становится очевидным, что не все сотрудники этой организации могут или должны знать все обо всей инфраструктуре. Кроме того, просто непрактично (и на самом деле, даже невозможно), чтобы один человек принимал все каждодневные решения. В связи с этим начинают появляться следующие отделы.

- *Управление*: определение общих стратегий и руководство всеми остальными отделами.
- *Администрирование*: общение с клиентами по поводу их потребностей, заключение контрактов, рассылка счетов, размещение заказов, осуществление выплат поставщикам, набор и увольнение сотрудников.
- *Разработка*: проектирование, реализация и тестирование новых продуктов и служб перед их предоставлением клиентам.
- *Производство*: непосредственное предоставление услуг клиентам как виде системного управления, так и виде поддержки и обучения конечных пользователей. Здесь можно выделить еще два отдела: операционный отдел (работа с компьютерами) и отдел поддержки (работа с людьми).

Эти функциональные подразделения не обязательно должны закрепляться за отдельными людьми. Даже при наличии одной небольшой системы и одного системного администратора они могут стать просто четырьмя разными обязанностями этого администратора.

Относительный размер каждого из этих отделов отражает тип организации, которую обслуживает административная группа. В исследовательском университете имеются организации самых разных типов, начиная от компании, занимающейся производством программного обеспечения, и заканчивая фирмой-изготовителем. Наш опыт работы в исследовательском университете показывает, что на управление должно приходиться 5% от всей работы, на администрирование — около 10%, на разработку — 10–20%, а на производство — соответственно, остальные 65–75%.

Мы полагаем, что в организациях небольшого и среднего размера все сотрудники должны хотя бы по несколько часов работать в отделе поддержки. Это позволит техническому персоналу получать информацию о проблемах, сложностях и пожеланиях клиентов из "первых рук". Далее эта информация сможет быть использована для улучшения текущей среды или проектирования будущих систем и служб. Кроме того, это также позволит клиентам периодически общаться с более опытными специалистами. (Вам когда-нибудь доводилось иметь дело с одноуровневой технической поддержкой?)

20% может показаться слишком высоким показателем для разработки, но в противном случае при увеличении количества и сложности поддерживаемых систем масштабировать имеющиеся решения будет не возможно. Исходные концепции управления будут накапливаться и постепенно устаревать без надежды на возрождение.

Идеальным числом людей, которым может непосредственно управлять один человек, считается от 5 до 10. Любое число выше этого потребует усложнения иерархии.

Все пять перечисленных категорий обязанностей более подробно описываются далее.

30.3. Роль управления

Главной задачей руководителя является упрощение работы технического персонала, состоящего из экспертов, которые выполняют "настоящую" работу в данной организации. На плечи руководителя ложится несколько обязанностей.

- Выполнение роли руководителя группы, определение направления и предоставление необходимых ресурсов.
- Набор, увольнение, оценка и развитие навыков персонала.
- Распределение заданий и контроль за ходом их выполнения.
- Руководство вышестоящим руководством.
- Решение проблем типа конфликтов между сотрудниками, недовольных пользователей, старого оборудования и т.д.
- Выполнение роли "высшей инстанции", к которой пользователи могут обращаться со своими проблемами.
- Контроль над разработкой масштабируемой инфраструктуры.
- Составление планов на случай аварии и других чрезвычайных обстоятельств.
- Извлечение документации из "заморочливых" голов системных администраторов.

Может показаться, что в этом списке не хватает такой обязанности, как общение с клиентами. Однако мы считаем, что эта роль больше подходит членам технического персонала. Руководителям обычно не хватает технических знаний для того, чтобы правильно оценить степень сложности и выполнимости требований клиента. Сюрпризов будет меньше для обеих сторон, если в определении сроков доставки и составлении гра-

фиков выполнения будут принимать участие именно те, кто и будет этим непосредственно заниматься.

Ниже все эти обязанности менеджера рассматриваются более подробно. Некоторые из них являются обязанностями не только руководителя, и потому будут описываться и в других разделах. Например, разработка масштабируемой инфраструктуры предполагает участие не только руководителя, но и разработчиков, поэтому эта обязанность будет также описываться и в разделе 30.5. Точно также восстановление после аварий — это задача не только руководителя, но и сотрудников отдела операций, и поэтому будет дополнительно рассматриваться в разделе 30.10. Документация является настолько важным и настолько проблематичным вопросом, что ей посвящен целый отдельный раздел — раздел 30.8.

Руководство

Руководство трудно описать. Но когда оно отсутствует или плохо осуществляется, это сразу же становится заметным. В некотором роде руководство — это "системное администрирование" организаций: оно тоже подразумевает выбор направления, проверку компонентов "на совместимость" и поддержание всей "системы" в рабочем состоянии с наименьшим возможным количеством "сообщений об ошибках".

К сожалению, технические навыки, позволяющие человеку стать замечательным администратором компьютерных систем, вовсе не обязательно означают, что он сможет хорошо справляться с ролью руководителя, которая требует скорее умения работать с людьми. Совладать с людьми гораздо сложнее, чем с языком Perl, например.

Новым руководителям с хорошими техническими навыками может быть особенно трудно сфокусироваться на выполнении роли руководителя и избежать соблазна заняться разработкой. Намного легче и интереснее погрузиться в решение технической проблемы, чем вести длительные беседы с "проблемным" сотрудником. Но что важнее для организации?

Проще всего определить свой уровень как руководителя следующим образом: составить список задач, над которыми сейчас работает организация, одним цветом выделить области, в которых всем руководите вы, а другим — области, в которых всем руководит кто-то другой, а вы лишь ему подчиняетесь, и потом посмотреть, какой цвет преобладает на получившейся картинке.

Набор, увольнение и управление персоналом

Управление персоналом может оказаться особенно трудной обязанностью. Выполняя руководящую роль, дело приходится иметь как с техническими, так и с личностными качествами сотрудников. Согласно стереотипу, системные администраторы, гениальные с технической точки зрения, как правило, совсем не умеют общаться и порой вообще лучше ладят с машинами, чем с людьми. Руководитель должен следить за кривой роста у сотрудников в обеих этих областях.

Технический рост стимулировать и оценивать довольно легко, но личностный рост не менее важен. Ниже приведены несколько важных вопросов, которые следует задавать при оценке "пользовательского интерфейса" сотрудника.

- Вписывается ли поведение этого человека в нашу рабочую среду?
- Как этот человек общается с начальством, клиентами и поставщиками?
- Ладит ли этот человек с другими членами команды?

- Имеют ли у этого человека задатки руководителя, которые следовало бы развить?
- Как этот человек реагирует на критику и технические споры?
- Старается ли этот человек устранить пробелы в своих знаниях?
- Каковы коммуникативные навыки этого человека?
- Может ли этот человек спланировать, реализовать и продемонстрировать проект клиента?

Набор персонала

Оценивать подобным образом следует не только уже работающих сотрудников, но и тех, что претендуют на получение места в данной организации. Личностные качества кандидатов часто не учитываются или недооцениваются. Не стоит быть здесь невнимательным, чтобы не пришлось жалеть позже!

Ответы на некоторые из этих вопросов, скорее всего, удастся получить при личной беседе. Но разговор по телефону с лицами, предоставившими рекомендацию, обычно позволяет узнать больше. Слушать нужно очень внимательно: многие не любят плохо отзываться о бывших подчиненных или коллегах и потому дают понять о наличии проблем у претендента при помощи очень тонких намеков, уловить которые можно, лишь внимательно слушая. Отсутствие у претендента рекомендаций с последнего места работы, должно настораживать.

Существует два подхода к формированию штата системных администраторов:

- нанимать опытных специалистов;
- обучать своих собственных.

Опытные специалисты обычно быстрее включаются в работу, но всегда есть что-то, от чего их нужно отучать. Для того чтобы выполнять свою работу, им необходим привилегированный доступ. Но пока о них известно мало, лучше, чтобы данные компании не попадали им в руки. В случае приема на работу опытных специалистов за ними, вероятно, придется "последить" до тех пор, пока не станет ясно, что им можно доверять.

Подготовка системного администратора требует определенного времени и усилий, да и компьютерная сеть производственного предприятия — не лучшее место для ее проведения. Но при наличии хорошего кандидата (умного, заинтересованного, внимательного и т.д.), конечный результат обычно более чем оправдывает затраты.

Некоторые качества хорошего системного администратора противоречат друг другу. Администратор должен быть достаточно дерзким для того, чтобы не боятся принимать неординарные решения при возникновении сложной проблемы, но при этом и достаточно осторожным для того, чтобы не нанести вред системе. Навыки в межличностных отношениях и решении проблем одинаково важны, хотя, по нашим наблюдениям, у многих администраторов они являются взаимоисключающими.

Мы разработали две методики оценки опытных претендентов. Раньше мы называли эти методики "тестами", но потом узнали, что в некоторых учреждениях (наподобие государственных университетов в США) не разрешается тестировать кандидатов. Поэтому теперь мы не "тестируем", а "оцениваем".

Первая методика заключается в использовании специальной письменной анкеты, предлагающей претендентам самостоятельно оценить свой опыт и знания в различных областях системного и сетевого администрирования. Возможные баллы — от 0 до 4:

- никогда не слышал об этом (0);
- слышал об этом, но не делал (1);
- делал это и могу делать это под контролем (2);
- могу это делать без постороннего контроля (3);
- хорошо разбираюсь в этом, могу даже обучать этому других (4).

В этой анкете есть вопросы-ловушки. Например, в разделе, посвященном аппаратным средствам, сразу же после вопроса о разъемах RJ-45 идет вопрос о "разъемах" MX (а ведь MX — это одна из записей DNS, а не тип сетевого или последовательного разъема). Такие вопросы позволяют оценивать степень "угадывания" в ответах претендента. Если претендент поставил напротив вопроса о "разъемах" MX бал 3, к нему следует отнестись с подозрением. В личной беседе после письменного анкетирования вовсе не помешает задать ему с невинным видом следующий вопрос: "Так для чего же вы используете разъемы MX?".

Вторая методика тоже заключается в получении ответов на ряд вопросов, но только не в письменном виде, а по телефону, и удобна при проведении собеседований о приеме на работу по телефону. Вопросы сформулированы так, чтобы быстро на них могли ответить только те, кто действительно разбирается в своем деле. За правильный ответ дается +1 балл, за ответ "я не знаю" — 0 баллов, а за явное угадывание или поиск ответа в руководствах — -1 балл. За разумный ответ типа "я не знаю, но вот что бы я сделал, чтобы узнать" тоже можно давать +1 балл. Нередко, умение отыскать нужную информацию является столь же важным, как и знание всех деталей наизусть.

Для нас результаты опросов по этим двум методикам всегда были достаточно надежным показателем. Количество вопросов-ловушек зависит от конкретной ситуации и требований сотрудников отдела кадров; как правило, одного-двух таких вопросов недостаточно для того, чтобы снять того или иного кандидата "со счетов".

Увольнение персонала

Допустив ошибку при подборе сотрудника, лучше уволить его как можно раньше. Промахи с подбором персонала случаются, но держать на работе людей, не желающих тянуть общую лямку, — значит настраивать против себя других сотрудников, потому что в таком случае именно им придется взваливать на свои плечи обязанности этих лентяев и подчищать за ними их огрехи. Клиенты тоже быстро заметят, что тот-то и тот-то не делают то, чего от них просят, и начнут требовать, чтобы их задания выполнял какой-нибудь конкретный системный администратор. Мало какому руководителю понравится давление со стороны клиентов и принятие ими вместо него решений по поводу того, кому какое задание следует дать.

Во многих организациях, однако, уволить кого-либо очень трудно, особенно по окончании испытательного срока, поэтому к первоначальной оценке следует относиться серьезно. Не исключено, что позже все-таки придется собирать факты, подтверждающие некомпетентность сотрудника, делать ему официальные выговоры, устанавливать рабочие нормы и т.д. В особо сложных случаях, единственным способом уволить некомпетентного, но сообразительного сотрудника может оказаться сокращение его должности.

Тщательное тестирование и контроль качества выполняемой работы

Именно руководитель определяет, что значит выполненная задача. Неопытные системные администраторы часто считают, что проблема устранена, и снова получают еще несколько отчетов о неполадках, тем самым более медленно, но вместе с тем качественно и правильно выполняя поставленную задачу. Такой сценарий может иметь место, например, из-за того, что первый пользователь, который сообщил о проблеме, описал ее неточно, или, возможно, из-за того, что какой-то из пользователей предложил неадекватное (или неверное) решение, а системный администратор просто взял и принял его, не потрудившись разобраться во всем самостоятельно. Так же не менее часто такое происходит и потому, что системный администратор не удосужился тщательно протестировать найденное решение.

Ниже перечислены некоторые основные ошибки:

- для нового программного обеспечения нет `man`-страниц и документации;
- программное обеспечение инсталлировано не везде, где нужно;
- права на использование программного обеспечения принадлежат только администратору или просто вообще установлены неправильно.

Тестирование, конечно, утомляет, но, пропустив этот важный этап, администратор может вдвое снизить производительность системы. Каждое сообщение о неисправности неизбежно приводит к трате времени и усилий — как пользователей, так и администратора[1]. Задание считается невыполненным до тех пор, пока не будут устранены все эксплуатационные шероховатости и пока клиент не останется доволен.

Обычно бывает так: пользователь сообщает, что у него на компьютере Y не работает команда или программа X; администратор отправляется к компьютеру Y, пробует выполнить команду X и у него она работает нормально. Здесь важную роль играет фраза “у него”. Попробуй он на самом деле выполнить эту команду от лица того пользователя, который сообщил ему о проблеме (например, путем выполнения перед ней команды `udo su - `*`имя_пользователя`*), проблема могла бы быть выявлена с первой же попытки. Аргумент “-” после `su` вынуждает результирующую оболочку использовать среду указанного пользователя, т.е. позволяет воспроизвести именно ту среду, о которой сказали, что она не работает.

В некоторых случаях полезно выполнять тестирование в несколько этапов. Например, при подозрении, что источником проблемы P является агент X, следует попробовать удалить агента X и посмотреть, исчезнет ли проблема P. Затем, нужно снова добавить агента X и посмотреть, появится ли проблема P опять. После этого, можно окончательно удалить X и еще раз удостовериться в том, что проблема P исчезла. При таком подходе у вас будет целых два шанса заметить, если P и X на самом деле никак не связаны друг с другом.

Пользователи обычно огорчаются, когда проблему не удается полностью решить с первой попытки, поэтому лучше не обманывать их ожиданий. Совсем не помешает пригласить пользователя, сообщившего о проблеме, поработать вместе над ее решением, особенно если проблема связана с незнакомым пакетом программ. Это позволит вам, как администратору, получить дополнительную информацию, а пользователю — изменить свое мнение о том, что ваше участие является исключительно консультативным.

[1] Иногда процедура диагностики или обращение в службу поддержки стоит немалых денег. Например, на текущий момент внешняя поддержка, как правило, обходится в приблизительно 75 долларов за каждый вызов специалиста или новую процедуру диагностики.

Управление, а не вмешательство

Технически подкованному руководителю может постоянно хотеться посоветовать сотрудникам, как они должны выполнять свою работу. Но поступая подобным образом, он может лишить их шанса приобрести свои собственные навыки и стать полностью ответственным за их работу. Даже еще хуже, если он будет не только советовать, но и также принимать решения по поводу того, какие задачи должны быть выполнены. Такое дотошное управление неизбежно закончиться недовольством сотрудников, лишенных возможности пользоваться своим собственными знаниями и творческими качествами.

У тех, кто действительно знает все лучше своих сотрудников, что-то не так либо с организацией (возможно, она охватывает слишком узкую сферу деятельности), либо с должностью (возможно, эта должность им не подходит), либо с персоналом. Еще один вариант — помешательство на власти.

Награда за хорошее поведение

В университетах преподаватели могут брать так называемый "творческий отпуск" — полгода или год, на протяжении которого нет никаких занятий и можно отправиться в какой-нибудь другой университет для проведения исследований или работы над новыми проектами. Смысл таких отпусков очевиден: для того, чтобы преподаватели работали эффективно, они должны встречаться с другими преподавателями и посещать другие университеты.

Системные администраторы тоже нуждаются в творческом отпуске, но часто их находят столь незаменимыми, что не могут отпустить на такой длительный срок, т.е. до тех пор, пока они не устанут и, в конце концов, не уйдут сами. Обмен системными администраторами с другой организацией на лето (или даже более долгий срок) может стать наградой для ценных сотрудников и позволить им немного снять напряжение от постоянной работы "на передовой". "Позаимствованным" системным администраторам, возможно, понадобится какое-то время на то, чтобы разобраться в локальной инфраструктуре, но они принесут с собой из своей организации новые идеи и решения, которые могут улучшить локальные стратегии системного администрирования.

В случае привлечения к такому проекту по обмену иностранных организаций может понадобиться разработать специальную схему финансирования, чтобы обойти волокиту с оформлением официальных рабочих виз. Мы несколько раз проводили подобный обмен на условиях, что приехавшие по обмену системные администраторы получают зарплату от своих организаций. В одном из этих случаев, администраторы обменивались квартирами, машинами, домашними животными, и т.д. — в общем, всем, ну или почти всем...

Назначение заданий и контроль за их выполнением

Одним из самых важных моментов в управлении проектом является проследить за тем, чтобы за выполнение каждой задачи отвечал какой-нибудь один определенный человек.

Избегать нужно двух вещей:

- невыполнения заданий из-за недоразумений, вызванных тем, что каждый думает, что их выполняет кто-то другой;
- излишней траты ресурсов из-за дублирования усилий, которое происходит в случае, когда множество людей или групп работают над одной и той же проблемой и их действия никем не координируются.

Работа может выполняться совместно, но, как показывает наш опыт, ответственность лучше не распределять между несколькими людьми, т.е. лучше делать так, чтобы за каждую задачу отвечал только какой-то один конкретный человек. Этим человеком вовсе не обязательно должен быть руководитель — им может быть любой, кто согласен выполнять роль координатора или руководителя проекта.

Важным преимуществом такого подхода является незаметное обеспечение ясности по поводу того, кто что реализовывал и кто какие изменения вносил. Эта ясность становится особенно важной тогда, когда возникает необходимость выяснить, почему то-то и то-то было сделано определенным образом или почему что-то вдруг начало работать по-другому или перестало работать вообще.

Быть "ответственным" за выполнение задачи вовсе необязательно означает быть "козлом отпущения" на случай возникновения проблем. В организациях, где на это смотрят именно так, количество доступных руководителей проектов будет быстро сокращаться. На самом деле весь смысл состоит просто в том, чтобы устранить неясность по поводу того, кто должен заниматься той или иной проблемой или вопросом.

С точки зрения клиента хорошей системой распределения заданий является система, в которой проблемы попадают "в руки" тем, кто в них лучше всего разбирается и может решить их быстро и качественно. С точки зрения руководителя, однако, назначаемые задания периодически должны быть на порядок сложнее уровня знаний того, за кем они закрепляются, для того, чтобы кадры могли продолжать расти и совершенствовать свои навыки по ходу работы. Т.е. задачей руководителя является отыскать баланс между необходимостью использования сильных сторон сотрудников и необходимостью совершенствования их знаний, причем так, чтобы довольны были все — и клиенты и сотрудники.

Для распределения повседневных административных задач больше всего подходит диагностическая система. Такие системы гарантируют, что сообщения (отчеты) о неполадках не останутся без ответа и будут незамедлительно направлены человеку, который разбирается в данной проблеме лучше всего. Диагностическим системам посвящен отдельный раздел 30.9, поэтому за дополнительной информацией следует обращаться именно туда.

Большие задачи могут быть разными и даже представлять собой целые проекты по разработке программного обеспечения. Они могут требовать использования формальных средств управления проектом и четких принципов разработки программного обеспечения. Здесь эти средства не описываются, но, тем не менее, они очень важны и должны учитываться.

Иногда системные администраторы знают, что необходимо выполнить определенную задачу, но не выполняют ее, потому что она неприятна. Сотрудник, указавший на пропущенную, не назначенную или не популярную задачу, скорее всего, получит ее в качестве задания. Такая ситуация приводит к конфликту интересов, потому что побуждает сотрудников "помалкивать" в подобных случаях. Не допускайте подобного в своей организации.

Руководство вышестоящим начальством

Чтобы эффективно выполнять свои обязанности, руководителю нужно, чтобы его уважало и поддерживало начальство. Ему необходимо иметь возможность определять структуру и штат своих сотрудников, а также принимать решение о том, кого принять на работу, а кого уволить. Ему нужно контролировать процесс распределения заданий,

а также право решать, когда поставленные цели достигнуты и когда сотрудники могут получить новые задания. И, наконец, ему необходимо быть ответственным за представление своей команды как внутри своей собственной организации, так и в мире в целом.

Вышестоящее руководство часто не имеет ни малейшего представления о том, чем занимаются системные администраторы. Предоставляйте им такую информацию, пользуясь возможностями своей диагностической системы, когда хотите убедить начальника в необходимости найма дополнительных сотрудников или приобретении нового оборудования.

Фиксировать информацию о выполняемой системными администраторам работе может быть полезно и даже при отсутствии конкретной цели. Руководители, особенно менеджеры нетехнического звена, часто недооценивают сложность и трудоемкость стоящих перед системными администраторами задач. Особенно это касается задач, связанных с поиском и устранением неисправностей.

Старайтесь подходить к планированию реалистично. Если не уверены, удваивайте или утраивайте время, которое, на ваш взгляд, необходимо для решения масштабных и очень важных задач. Если обновление системы будет выполнено за два дня вместо трех, пользователи только поблагодарят вас вместо того, чтобы ругать, как они бы сделали, если бы вы решили, что оно может быть выполнено за один день.

Условия лицензирования программного обеспечения часто не совсем соответствуют фактическому объему использования лицензионного программного обеспечения в компании; вышестоящее руководство имеет склонность закрывать на это глаза. О том, как можно сгладить подобные несоответствия между стратегиями компании и реальностью, более подробно будет рассказываться чуть позже в этой главе, в разделе “Лицензии на программное обеспечение”.

Труднее всего обычно получить поддержку руководства в вопросах ужесточения стратегии безопасности. Чем жестче правила безопасности, тем больше неудобств испытывают пользователи, а поскольку их больше, их протесты быстро находят отклик у начальства. Увеличение степени безопасности может снижать продуктивность пользователей, поэтому прежде чем вносить то или иное изменение в стратегию безопасности, следует проводить анализ степени риска, дабы быть уверенными в том, что руководство и пользователи понимают, почему оно предлагается.

Также нужно следить за тем, чтобы пользователи были заранее оповещены обо всех планирующихся изменениях в стратегии безопасности, которые могут повлиять на их работу (например, о переходе с паролей на ключи RSA/DSA для удаленной регистрации). Все эти изменения должны быть хорошо документированы и сопровождаться поддержкой во время внедрения. Документация должна быть понятной и содержать пошаговые инструкции по работе с новой системой. При переходе на новую систему лучше выделить дополнительное время на обслуживание пользователей, которые могут начать паниковать из-за того, что не успели прочитать свою электронную почту.

Разрешение конфликтов

Некоторые из ложащихся на плечи руководителя обязанностей связаны непосредственно с умением ладить с людьми (как правило, клиентами или кадрами) в напряженных ситуациях. Мы сначала рассмотрим, как следует вести себя в общем, а затем поговорим о таком особом случае, как работа с “норовистыми” клиентами, которых иногда еще также называют ковбоями.

В мире системного администрирования конфликты чаще всего возникают между системными администраторами и их клиентами, коллегами или поставщиками. Например, клиент может быть не доволен качеством предоставленных ему услуг, поставщик мог не предоставить вовремя обещанный материал, коллега мог сделать не то, что его просили, а конструкторский отдел — требовать полного контроля над установленными на их компьютерах конфигурациями ОС.

Посредничество

Большинство людей не любят говорить о конфликтах или даже признавать, что они существуют. Когда эмоции уже бушуют, это обычно означает, что конфликт распознан слишком поздно и что негатив накапливался на протяжении длительного периода времени. За это время стороны могли уже затаить серьезную обиду и не раз прокрутить в своей голове "подлые" намерения друг друга.

Встреча лицом к лицу под контролем нейтрального посредника иногда позволяет разрядить обстановку. Попробуйте ограничить спор одним вопросом, а выделенное на его обсуждение время — 30 минутами. Такие меры снижают вероятность превращения встречи в бесконечную "схватку".

Определив проблему и пожелания каждой стороны, попробуйте достичь соглашения так, чтобы избежать повторного возникновения проблемы. Воплотив решение в жизнь, периодически проверяйте ситуацию, дабы убедиться в его эффективности и долгосрочности.

Норовистые пользователи и отделы

Процесс внедрения строго контролируемых систем часто приводит к возникновению конфликта. Технические пользователи (а иногда и целые отделы) могут посчитать, что централизованное системное администрирование не может адекватно удовлетворять их конфигурационные потребности или их необходимость в автономном контроле над используемыми ими компьютерами.

Первым импульсом руководителя может быть просто взять и заставить этих норовистых пользователей принять стандартную конфигурацию для того, чтобы свести к минимуму затраты и время, требующиеся на их поддержку. Однако такой жесткий подход обычно заканчивается недовольством как среди пользователей, так и среди системных администраторов. Не забывайте, что желания норовистых пользователей зачастую бывают вполне законными и что именно системным администраторам приходится поддерживать их или, по крайней мере, воздерживаться от усложнения им жизни.

Лучше всего в такой ситуации выяснить причины, по которым норовистые пользователи не хотят принимать управляемые системы. В большинстве случаев их потребности можно удовлетворить и тем самым поставить их на место.

В качестве альтернативного варианта можно согласиться на автономию. Разрешите своим норовистым пользователям или отделам делать все, что им хочется, на условиях, что они сами будут отвечать за поддерживание своих систем в рабочем состоянии. Установите брандмауэр, чтобы защитить контролируемые вами системы от возможных проникновений или вирусов со стороны сети этих норовистых пользователей. И не подчищайте за ними никакие их огрехи.

Обязательно заставьте всех представителей "отделившейся" сети подписать соответствующий стратегический документ, оговаривающий определенные правила безопасности: например, что если их системы начнут мешать работе организации, они будут отключены

от общей сети до тех пор, пока на них не будут установлены необходимые заплаты и обновления и они не перестанут негативно влиять на производственный процесс.

В случае постоянного увеличения количества приносимых сотрудниками на работу переносных компьютеров, без творческого подхода к системному администрированию не обойтись. Доведется придумать, как обслуживать эти (потенциально ненадежные) устройства, не подвергая опасности целостность имеющихся систем. Отдельная сеть может оказаться неплохим вариантом.

В абсолютно всех организациях есть так называемые "пользователи-передовики", которые любят устанавливать все самые последние новинки сразу. Такие пользователи не чураются ни бета-версий, ни нестабильных предварительных выпусков программных продуктов; им главное, чтобы у них было самое новое программное обеспечение. К таким пользователям следует научиться относиться как к полезным ресурсам, а не как к "бельму на глазу". Они являются просто идеальными кандидатами для тестирования нового программного обеспечения и зачастую с удовольствием составляют отчеты об обнаруженных в нем дефектах, тем самым предоставляя возможность вовремя устранить проблемы.

30.4. Роль администрирования

В данном контексте под "администрированием" подразумеваются все задачи, которые не имеют прямого отношения к выпускаемой организацией продукции, такие как бухучет, учет трудовых ресурсов, секретарская работа, продажи, закупка и другие связанные с материально-техническим снабжением процессы. Также оно довольно значительно отличается от "администрирования" в "системном администрировании".

В зависимости от организации, административные задачи могут полностью выполняться либо в самой организации, либо в родительской организации, либо половина на половину. Но все они где-нибудь да должны выполняться.

В небольших автономных группах эти задачи чаще всего распределяются между различными штатными сотрудниками. Однако такой подход никак не назовешь эффективным использованием высококлассных и встречающихся столь редко IT-специалистов. Кроме того, многие люди ненавидят административную работу. Правовые и организационные требования заставляют следовать строгим правилам, которые не всегда устраивают предпочитающих творческий подход профессионалов. Задачей руководителя является убедиться в том, что административные задачи передаются "наверх" и выполняются специально нанятым для этого персоналом или, по крайней мере, равномерно распределены между всеми системными администраторами.

Мы рассмотрим здесь лишь несколько административных задач, потому что большинство из них диктуется родительской организацией и не подлежит обсуждению.

Продажа

Принимая решение о том, чтобы взяться за проект для нового или существующего клиента, нужно посоветоваться с сотрудниками-специалистами по продажам, которые понимают потребности потенциальных клиентов, знают, какие услуги и продукты может предоставить ваша команда, и в курсе ее степени занятости. Нет ничего хуже, чем давать обещания, которые не смогут быть выполнены. Если в переговорах принимает участие начальник, нужно позаботиться о том, чтобы технические специалисты его как следует проинформировали. Более правильный вариант — позволить техническим специалистам вести переговоры, а начальник пусть выступает в роли наблюдателя.

Закупка

Во многих организациях команда системных администраторов и отдел снабжения работают отдельно. Нет ничего хорошего, если отдел снабжения принимает решения, не проконсультировавшись с системными администраторами. Даже еще хуже, если решения о закупке принимаются высшим руководством на поле для гольфа или на предоставленных поставщиком местах на стадионе во время первой в этом сезоне игры его любимой футбольной команды.

Вот одна из типичных историй: вице-президент IT-отдела компании-поставщика ленточных накопителей думал, что знает, какие накопители нужны для большого серверного проекта. В воскресенье, во время партии в гольф, он подписал контракт с IBM, не проконсультировавшись со своими системными администраторами. Дисковые массивы IBM предстояло подключать к серверу Sun, и представитель отдела продаж IBM клялся, что все будет работать нормально. Оказалось, что необходимые драйверы устройств еле-еле работали и прошли только стадию альфа-тестирования. В результате критически важный сервер компании пребывал в нерабочем состоянии гораздо более длительное время, чем планировалось, а именно — на протяжении целых трех следующих месяцев, пока IBM доводила драйверы "до ума". Установленные сроки сдачи проекта прошли, и сотрудники отдела IT-технологий выглядели для всех остальных полными идиотами. Простой в связи с установкой устройств хранения повлиял и на итоговый показатель прибыли компании — с первого места в своем сегменте рынка он опустилась на самое последнее, потому что опоздала с выпуском критически важных продуктов.

Системные администраторы могут предоставлять полезные сведения об имеющихся в локальной среде требованиях к совместимости, о компетентности поставщиков (особенно сторонних посредников) и надежности определенных типов оборудования. Информация о надежности играет критически важную роль в мире ПК, где цены настолько конкурируют между собой, что качество зачастую просто вообще игнорируется.

Системные администраторы должны уведомляться о предстоящих заказах любого нового оборудования для того, чтобы они могли определить, как его можно будет интегрировать в текущую инфраструктуру и какие проекты и ресурсы будут необходимы для его поддержки.

Обратите внимание на то, что в обязанности системных администраторов не входит принятие решений по поводу того, *может ли* поддерживаться та или иная система. Вы, конечно, можете давать рекомендации своим клиентам, но если для выполнения их задания требуется определенное оборудование, ваша задача — выполнить его. Возможно, для этого вам придется нанять дополнительных людей или остановить работу над другими проектами. Предоставьте начальству список вариантов и позвольте им там самим решать, какой компромисс их устраивает больше.

Участие системного администратора в спецификации приобретаемых систем особенно важно в тех организациях, где все покупается по минимальным ценам (таких как, например, государственные учреждения и университеты). В большинстве случаев при закупке разрешается указывать критерии оценки. Не стоит забывать включать и такие пункты, как "должна быть совместима с существующей средой" или "должна позволять нормально работать с пакетом программ XYZ".

Показатель воздействия и стоимости дополнительного компонента оборудования (или, иногда, программного обеспечения) не постоянен. Является ли он шестидесятым в этой архитектуре, или же первым? Достаточно ли у него места на жестком диске для системных файлов? Хватит ли у него памяти для выполнения современных объемных

приложений? Имеется ли свободный сетевой порт, к которому его можно подключить? Установлена ли на нем совершенно новая операционная система?

Такие вопросы обычно подразумевают под собой другой более фундаментальный вопрос: следует ли остаться верным традиции и приобрести оборудование у своего текущего поставщика или стоит попробовать модную новинку, предлагаемую недавно появившейся компанией, которая может быть потрясет весь мир, а может прекратит свое существование уже через год. Ответ на этот вопрос зависит от того, в какой организации вы работаете. Простым "да" или "нет" здесь не обойтись; зачастую приходится делать сложный выбор между новейшим навороченным оборудованием и машинами, которые уже хорошо знаешь, и к которым уже привык.

При наличии возможности переговорить с поставщиками (официально или каким-нибудь другим образом) системный администратор обычно может добиться более выгодных условий, чем отдел снабжения. Не стесняйтесь называть цены на аналогичное оборудование других поставщиков и нескромно давать надежду на возможное увеличение объема закупок в следующем году. Ведь представители отдела продаж, в конце концов, не скромничали, когда описывали достоинства своего продукта.

Делая закупки в конце квартала или непосредственно перед или после изменения линейки продуктов поставщика, можно добиться хороших скидок. Продавцам часто бывает нужно улучшить показатели объема продаж за квартал или побыстрее сбыть залежавшиеся на складе модели, которые вот-вот устареют.

Согласно нашему опыту, отыскать поставщиков с технически подкованными сотрудниками, к которым в любое время можно обратиться за поддержкой или консультацией, довольно сложно. Если вам удастся отыскать одного такого, постарайтесь с ним не ссориться.

После сделки человек, составлявший исходный заказ, должен проверить пришедший товар, упаковочный лист и, возможно даже, предоставленный поставщиком счет-фактуру, дабы убедиться в том, что пришло именно то оборудование, которое было оплачено. В большинстве организаций согласно установленной политике приобретенный продукт также должен быть занесен в локальную систему контроля инвентарными запасами.

Бухучет

Держать в порядке финансы — это еще одна важная административная задача. Однако условия ее выполнения настолько зависят от организации, что рассмотреть ее подробно здесь мы просто не можем. Что она подразумевает во всех случаях, так это внимательное изучение правовых и организационных требований и регулярное обновление записей. Бухгалтерские проверки бывают очень неприятными.

Кадры

Мы уже рассматривали вопросы приема сотрудников на работу и их увольнения в разделе 30.3, но для того, чтобы внедрить нового сотрудника в свою инфраструктуру, требуется гораздо больше, чем просто написать письмо с приглашением на работу. Нужно знать и следовать установленным в организации правилам касательно объявлений о вакансиях, испытательных сроков, проверок и т.д. К отдельной категории административных задач относятся процедуры предоставления новому человеку рабочего стола, компьютера, ключей, учетных записей, доступа `sudo` и т.д.

Возможно, даже более важным является сворачивание всех этих процессов, когда сотрудник сообщает о своем намерении уйти. В США о своем уходе принято сообщать за две недели. В некоторых организациях (таких как те, у которых работа связана с обеспечением безопасности) не ждут двух недель, а сразу же провожают сотрудника до дверей и лишают его всех прав на физический и сетевой доступ. Это время может быть использовано для проведения быстрой проверки того, не оставил ли увольняющийся сотрудник каких-нибудь потайных путей доступа к серверам организации. Некоторые коммерческие продукты (такие как служба каталогов производства Novell) позволяют чуть ли не мгновенно удалять все данные бывших сотрудников.

Когда планируется массовое сокращение кадров, системных администраторов обычно никто не спрашивает, хотя следовало бы. Настает день сокращений, и руководство начинает кричать, чтобы данные на компьютерах бывших сотрудников были сохранены в виде резервных копий, чтобы их доступ был отменен, и т.д. Рассерженные сотрудники незаметно забирают с собой оборудование, а все потому, что этот процесс был плохо спланирован.

Маркетинг

Системное администрирование — это интересная штука. Если ты выполняешь свою работу хорошо, пользователи воспринимают поддерживаемую тобой бесперебойную вычислительную среду как должное и не замечают, что ты вообще делаешь. Однако в современном мире вирусов, спама, объемных приложений и полной зависимости от Internet, IT-специалисты необходимы в любой организации.

Довольные клиенты являются наилучшим средством маркетинга для системного администратора. Однако существуют и другие способы стать "заметным" в своей организации и в более широких кругах. Исходя из своего личного опыта, мы предлагаем следующие методы как наиболее эффективные.

- Проводить регулярные встречи (например, в здании муниципалитета), на которых пользователи могли бы рассказывать о своих проблемах и задавать вопросы о компьютерах. К таким встречам можно готовиться, анализируя поступающие в службу поддержки вопросы от пользователей, и начинать их с короткой презентации самых волнующих тем. Чтобы публике было интересно, и она приходила снова и снова, темы следует периодически обновлять.
- Оставлять как можно больше времени на вопросы пользователей и иметь в распоряжении способных ответить на них специалистов. Не стоит "выкручиваться", если пользователь задал какой-то неожиданный вопрос. Если не можете дать точного ответа сразу, лучше признайтесь в этом и дайте ответ позже.
- Организовывать семинары либо для своих коллег-администраторов, либо для работающих в организации пользователей. Эти семинары лучше проводить с периодичностью раз в два или три месяца и сообщать о темах, которые на них будут рассматриваться, заранее[2].
- Посещать конференции по системному администрированию и читать лекции или писать доклады о разрабатываемых вами средствах. Это не только позволит получать отзывы от коллег, но и также будет показывать вашим клиентам (и начальству), что вы хорошо справляетесь со своей работой.

[2] Также можно приглашать на них системных администраторов "извне", чтобы они проводили свои презентации по имеющимся у них темам, главное проверять, чтобы эти презентации не отличались по качеству от тех, что проводят внутренние докладчики.

Хоть мы и перечислили эти маркетинговые технологии в посвященном администрирования разделе, вполне очевидно, что в маркетинге должна принимать участь вся команда, а не только администраторы.

Системное администрирование заключается в общении с людьми и решении их проблем. Личные отношения здесь важны так же, как и в любом другом деле. Общайтесь со своими клиентами и коллегами и оставляйте время на личные встречи и обмен информацией с ними.

При наличии большого количества клиентов лучше закрепить за каждым из них какого-то определенного человека, чтобы тот выступал в роли так сказать "персонального менеджера". Эта роль должна означать обеспечение общей удовлетворенности клиента и регулярное общение с его конечными пользователями. Передача новостей и информации об изменениях в компьютерной среде через этого человека будут создавать дополнительные возможности для контакта.

Дополнительные обязанности администратора

Администраторов часто заставляют выполнять множество дополнительных мелких поручений, примеры которых перечислены ниже.

- Вести учет продуктов (т.е. заниматься инвентаризацией).
- Приобретать мебель и другое оборудование для офиса.
- Следить за рабочим состоянием автомата для кофе и поставкой кофе, сахара и молока. Поставки кофе должны оплачиваться из бюджета организации для гарантии, что каждый сотрудник, подкрепившись кофе, будет работать в полную силу. Некоторые компании также запасаются легкими закусками и напитками.
- Поливать растения и цветы.
- Договариваться с отделом материально-технического обеспечения (или внешними подрядчиками) об уборке, обслуживании вентиляционной системы и т.д.
- Заботится о ключах или других средствах доступа.

30.5. Роль разработки

Большим организациям нужна надежная, масштабируемая компьютерная инфраструктура для того, чтобы системным администраторам не доводилось тратить все свое время на борьбу с жесткой, плохо спроектированной и требующей приложения множества ручных усилий системой. На практике это означает, что им необходимо специальное программное обеспечение, архитектурная структура и персонал, которому можно официально поручить заняться их разработкой и сопровождением.

Увеличение размеров организации и количества используемых в ней компьютеров со временем потребует поиска способов оптимизации и систематизации административного процесса. Наиболее распространенным (и наиболее мощным) средством оптимизации является автоматизация. Она сокращает объем ручного труда и обеспечивает последовательное выполнение задач.

К сожалению, эта автоматизация часто выполняется неструктурированным образом с таким печальным последствием, как медленное накопление в организации целой кучи быстрых исправлений и сценариев, которые постепенно превращаются в настоящий существующий сам по себе недокументированный кошмар.

По поводу того, до какого количества компьютеров в организации можно не беспокоиться о средствах оптимизации, постоянно ведутся споры. Мы считаем, что лучше не компьютеры считать, а думать о профессионализме и удовлетворенности работой. Постоянные проблемы и расстройства никак не делают жизнь веселее и, естественно, сказываются на качестве обслуживания. Требования тоже бывают разными: то, что подходит для системы, предназначенной для обслуживания биржевых маклеров, не подходит для студенческой лаборатории в университете, и наоборот. Решения должны соответствовать потребностям пользователей.

Для описания высокоуровневой концептуальной работы системных администраторов, в отличие от повседневных обязанностей, иногда используется такой замысловатый термин, как "архитектура инфраструктуры". Слово "инфраструктура" здесь означает, что IT-отделы стали восприниматься как полезная вещь или средство. Инфраструктура, подобно телефонной системе или водопроводным трубам, должна просто работать. В случае появления в ней "бреши" недостающий компонент должен быть быстро обнаружен и заменен. А слово "архитектура" здесь означает, что инфраструктура предназначена для конкретной цели и организована так, чтобы соответствовать потребностям своих клиентов.

В этом разделе рассматриваются лишь некоторые из возможностей функциональной архитектуры инфраструктуры. Более подробную информацию можно найти в источниках, которые перечислены в конце этой главы. А в главе 11 приводились полезные ссылки по вопросам внедрения.

Архитектурные принципы

В следующих разделах представлены перечни наилучших архитектурных решений, которыми следует пользоваться при планировании структуры IT-отдела в своей организации. Эти принципы особенно важны, когда конфигурация, которая будет поддерживаться, является новой или необычной, потому что примеры таких ситуаций может быть очень сложно отыскать в реальном мире. Все хорошо спроектированные процессы подразумевают соблюдение этих или близким к ним принципов.

Процессы должны быть воспроизводимыми

Системное администрирование — это не показательное выступление. Все, что делается, должно делаться последовательно и многократно. Обычно это означает, что низкоуровневые изменения должны вноситься сценариями или конфигурационными программами, а не самими системными администраторами. Расхождения в конфигурации должны перехватываться и фиксироваться в конфигурационных файлах соответствующего предназначенного для администрирования ПО.

Например, сценарий, настраивающий новый компьютер, не должен задавать вопросов о номерах IP-адресов и пакетах, которые нужно установить. Вместо этого он должен проверять каталог конфигурации системы и извлекать всю необходимую информацию оттуда. Он может отображать эту информацию для подтверждения, но выбираемые параметры должны быть предопределены. Чем меньше будет участвовать в этом процессе пользователь, тем меньше шансов, что он допустит какую-нибудь ошибку.

Но давайте уточним: мы не описываем здесь организацию, где все стратегические решения принимаются администраторами "высшего ранга" и беспрекословно выполняются "глупыми трутнями". Воспроизводимость не менее важна и тогда, когда вы являетесь единственным администратором в своей организации. Незапланированных конфигура-

ционных решений, не оставляющих никаких данных аудита, лучше не принимать. Если требуется что-то изменить, изменяйте центральную конфигурационную информацию и распространяйте изменения уже оттуда.

Оставлять отладочные данные

Кто сделал, что сделал и зачем? В случае появления проблем в системе, наличие возможности вернуться к последнему рабочему состоянию или, по крайней мере, выяснить, что изменилось с тех пор, позволит устранить их быстрее. Важно знать не только "что" изменилось, но и "кто" внес это изменение и "почему". Беседа с человеком, реализовавшим приведшее к возникновению проблем изменение, часто позволяет увидеть суть. Возможно, изменение удастся быстро отменить, но иногда бывает так, что изменение было внесено в благих целях, и его отмена может только ухудшить дело.

Системы контроля изменений являются одним из наиболее удобных способов для отслеживания изменений, о них более подробно рассказывалось в разделе 11.5. Они предоставляют как хронологический перечень фактических данных, так и информацию о том, каким из системных администраторов было внесено то или иное изменение. При правильном использовании каждое изменение сопровождается комментарием, в котором объясняется причина внесения данного изменения. Автоматизированные средства могут проверять конфигурационные файлы, которые они изменяют, и указывать на себя в этих комментариях. Благодаря этому выявить работающий неправильно сценарий и отменить внесенные им изменения, совсем нетрудно.

Еще одним полезным подходом является введение так называемого "дневника" электронной почты. Это может быть либо журнал, в который постоянно записываются сообщения, отправляемые работающим в данной организации администраторам, либо хранилище почты, в котором фиксируются только обновления дневника. Дневник представляет собой хорошо известный источник информации о подлежащих регистрации конфигурационных изменениях, а тот факт, что он реализуется посредством электронной почты, делает обновления не такими утомительными для системных администраторов. Недостатком дневника электронной почты является то, что информация в нем структурируется только в хронологическом порядке.

Осознавать важность документации

На самом деле документация является настолько важной для масштабируемой инфраструктуры, что мы посвятили ей целый отдельный раздел 30.8.

Настраивать и писать код

Использовать существующие средства — это хороший подход, и так следует поступать всегда, когда это возможно. Но двух абсолютно одинаковых организаций не бывает, и каждая организация обязательно имеет какие-нибудь уникальные требования. IT-инфраструктура, которая в точности удовлетворяет требования организации, увеличивает конкурентоспособность этой организации в целом и продуктивность ее сотрудников в частности.

Благодаря гибкости в плане написания сценариев и обилию средств с открытым исходным кодом, Linux является идеальной базой для создания хорошо отрегулированной инфраструктуры. С нашей точки зрения, группа системных администраторов, не умеющая выполнять такую функцию, как разработка ПО, является слабой.

Содержать систему в чистоте

Управление системами заключается не только в установке, добавлении и конфигурировании, оно также предполагает наличие знаний о том, что следует сохранить, что следует выкинуть и что следует обновить. Мы называем эту концепцию "устойчивым управлением". Иметь возможность добавить новый компьютер в среду за 5 минут и создать новую учетную запись пользователя за 10 секунд — это замечательно. Но если заглянуть в будущее, станет ясно, что наличие возможности отыскивать и удалять старые учетные записи и компьютеры организованным образом является не менее важным. Устойчивость в управлении системами означает, что у вас есть концепции и средства необходимые для того, чтобы вы могли выполнять свою работу организованным образом в течение длительного срока.

Анатомия системы управления

Система управления должна включать следующие основные элементы.

- **Автоматизированная процедура установки новых компьютеров**. Подразумевается не только установка операционной системы, но также и добавление дополнительного программного обеспечения плюс локальное конфигурирование, необходимое для того, чтобы компьютер можно было начать использовать в производственном процессе. В любой организации рано или поздно придется поддерживать несколько типов конфигурации, поэтому лучше с самого начало включать в свои планы множество типов машин. В разделе 11.1, посвященном инсталляции Linux, рассказывалось о нескольких системах, которые могут упрощать процедуру установки.
- **Систематическая установка заплат и обновлений на существующих компьютерах**. На случай возникновения в системе какой-нибудь проблемы должен существовать стандартизированный и простой путь внедрения на всех пострадавших компьютерах соответствующих обновлений. Обратите внимание на то, что поскольку компьютеры не находятся во включенном состоянии все время (даже если так задумано), схема обновления должна предусматривать способ обновления тех компьютеров, которые на момент инициализации процесса обновления не были подключены к сети. Можно выполнять проверку на наличие обновлений во время загрузки, а можно проводить процедуру обновления регулярно, по графику; подробнее об этом рассказывалось в конце раздела 11.6.
- **Система мониторинга**. Не должно быть так, чтобы пользователи звонили сами и сообщали о том, что сервер не работает. Это не только непрофессионально, но и не позволяет знать точно, насколько долго система находится в нерабочем состоянии. Первый позвонивший человек может быть далеко не первым, кто столкнулся с проблемами. Поэтому должна существовать какая-нибудь система мониторинга, которая будет подавать сигнал тревоги сразу же при появлении проблем. Однако схему подачи сигналов тревоги следует как следует продумать, потому что если сигналов тревоги будет слишком много, системные администраторы начнут просто игнорировать их, а если их будет слишком мало, незамеченными могут остаться какие-то важные проблемы.
- **Система связи**. Обязательно нужно наладить связь с пользователями: их поддержка является главной целью во всем, что делает системный администратор. Система отслеживания запросов является, можно сказать, предметом первой необходимости (подробнее о ней будет рассказываться в разделе 30.9). Центральный каталог, в

котором пользователи смогут просматривать информацию о состоянии и контактах и который обычно размещается в Web, тоже не помешает.

Набор инструментов системного администратора

Хорошие системные администраторы ленивы (сообразительны) и обычно стараются автоматизировать свою работу. Успешный отдел системных администраторов пишет сценарии для "заполнения пробелов" между средствами, приобретенными у производителей, и средствами, загруженными из Internet, и затем делает так, чтобы они были доступны всегда, когда в них возникает необходимость.

В большинстве организаций имеется свой собственный сценарий `adduser` и возможно даже также сценарий `rmuser`. Если вы являетесь единственным системным администратором, вы, скорее всего, сохраните эти два небольших "драгоценных" сценария в своем собственном каталоге `~/bin` и будете потом жить спокойно. Но если вы работаете в группе с двумя или более людьми, все становится сложнее. Временные меры, такие как копирование средств друг друга и изменение путей поиска так, чтобы они указывали на каталоги `~/bin` коллег, на какое-то время спасут. Однако со временем все равно придется искать более устойчивое решение.

Нашим решением для этой проблемы является "набор инструментов системного администратора" (sysadmin tool box). Этот набор включает все разработанные локально программы системного администрирования и размещается в своем собственном специальном дереве каталогов (например, `/usr/satools`). Более того, у нас есть специальный стратегический документ, в котором описано, как должны писаться программы для этого набора. "Владельцем" набора назначен один из системных администраторов, и именно он и отвечает за принятие любых стратегический решений, которые не предусмотрены в этом документе.

При создании такого стратегического документа важно четко обосновать присутствие в нем каждого пункта. Только понимая причину, люди смогут следовать духу стратегии, а не только ее буквальным правилам.

Ниже перечислены несколько возможных пунктов такого стратегического документа.

- Опубликованные программы-инструменты доступны в каталоге `/usr/satools/bin`. Конфигурационная информация хранится в каталоге `/usr/satools/etc`. Журнальные файлы и любые другие файлы данных, которые программы создают во время работы, помещаются в каталог `/usr/satools/var`, а файлы статистических данных — в каталог `/usr/satools/share`. Причина: для стандартизации.
- Программы должны писаться на языке Perl или Python. Причина: для стандартизации и для упрощения передачи и многократного использования их кода системными администраторами.
- У каждой программы должна быть своя собственная `man`-страница. Причина: в случае отсутствия документации у программы то, как она работает, известно только ее автору.
- Разрабатывая программу, сначала следует написать документацию и дать просмотреть ее тому, кто отвечает за набор средств системного администрирования, и только потом приступать к написанию кода. Причина: для выявления избыточности и стратегических несоответствий на ранней стадии.
- Каждая программа должна принимать опцию `--no-action`, показывающую, что будет происходить, если программа ничего не будет делать. Если реализовать та-

кую опцию невозможно, программа должна, по крайней мере, завершать свою работу с соответствующим сообщением. Причина: для предотвращения катастроф, особенно когда речь идет о более мощных программах, таких как `rmuser` или `mvhomedir`.

- Все программы должны добавляться в систему контроля изменений, а именно — в каталог `svn://server/satools` (репозиторий Subversion, см. раздел 11.5). Причина: над программой могут работать несколько человек, а система контроля изменений упрощает управление исходным кодом и устранение дефектов.

В реальном мире такой стратегический документ может состоять из нескольких страниц.

Кроме того, вы можете написать небольшую утилиту для управления самим набором инструментов системного администратора, которая будет устанавливать новые программы из репозитория Subversion в каталог `/usr/satools`. Для контроля качества эта утилита может также выполнять какую-нибудь проверку на предмет соответствия перечисленным в документе правилам. Например, она может контролировать, чтобы каждая программа устанавливалась с соответствующей документацией.

Принципы разработки программного обеспечения

Поскольку административные сценарии не всегда выглядят как "настоящее" программное обеспечение, вы можете решить, что продумывать их так же тщательно, как типичные проекты по разработке ПО, не нужно. В некоторых случаях это действительно так, а в некоторых — нет. При определенной степени сложности и критичности к таким проектам следует начинать относиться особенно серьезно.

Хорошим и удобным источником по этой теме является книга Фредерика Брукса-мл. (Frederick P. Brooks, Jr.) *The Mythical Man-Month: Essays on Software Engineering*. Интересно то, что проблемы разработки ПО, которые он описывал в 60-70 гг., актуальны до сих пор. Ниже перечислены некоторые основные принципы разработки ПО и встречающиеся в этом процессе ловушки (порядок значения не имеет).

- Проекты должны иметь четко определенный контекст и срок завершения.
- Обязанности должны быть четко распределены. Администратор, работающий над проектом, начальник отдела системного администрирования и конечный заказчик должны обговорить между собой все цели, рамки и потенциальные факторы риска.
- Необходимая подготовительная документация должна быть составлена до того, как начнется написание кода.
- Интерфейсы, необходимые для осуществления взаимодействия между различными компонентами, должны быть определены и документированы заблаговременно.
- Критические фрагменты кода должны сначала воспроизводиться в виде прототипа и тестироваться. Очень важно получить отзывы от людей, которые будут пользоваться данным ПО, на как можно более ранней стадии разработки.
- Включайте в каждый проект этап постустановочного анализа: анализируйте ошибки и достижения.

Оптимальные методики управления проектами по разработке ПО всегда были предметом горячих споров. За последние десять лет популярными в среде разработчиков стали так называемые "быстрые" (agile) методики, делающие акцент на ускоренном создании прототипов, раннем развертывании, постепенном добавлении функциональ-

ных возможностей и минимизации накладных расходов. Им еще далеко до того, чтобы называться универсальными, но многие идеи, похоже, не лишены здравого смысла, а их упрощенные версии прекрасно подходят для меньших по размеру проектов, встречающихся в области системного администрирования. Подробнее о них можно узнать по адресу `en.wikipedia.org/wiki/Agile_methods`.

30.6. Роль операций

Операционный отдел отвечает за установку и сопровождение IT-инфраструктуры. Вместе с отделом поддержки он является частью производственного отдела, как рассказывалось в разделе 30.2. Как правило, операционный отдел имеет дело с компьютерами и проводами, а отдел поддержки — с людьми.

Операционный отдел занимается только существующим службами; все "новинки" должны проектироваться, реализовываться и тестироваться отделом разработки и только потом внедряться в производственную среду.

Операционный отдел фокусируется на создании стабильной и заслуживающей доверия среды для клиентов. Доступность и надежность являются его ключевыми задачами. Сотрудники операционного отдела не должны ни проводить никаких экспериментов, ни делать никаких быстрых исправлений или улучшений по пятницам. Просто вероятность того, что это закончится появлением проблем (заметить которые на выходных смогут только клиенты) слишком высока.

Обеспечение минимального времени простоя

Многие люди зависят от предоставляемой нами вычислительной инфраструктуры. Факультет в университете, пожалуй, сможет какое-то время обходится без своего Web-сайта, а вот компания, принимающая заказы по почте через Internet, такая как Amazon.com — нет. Некоторые даже не заметят, что сервер печати не работает, но студента или профессора, которого поджимают сроки сдачи курсовой или заявки, это действительно очень расстроит. Даже в университетской среде потеря доступа к электронной почте обычно выводит всех из себя. Центральные файловые серверы тоже являются потенциальным источником неприятностей[3].

В некоторых организациях придется обеспечивать наличие "аварийной службы". В коммерческой среде это может означать непосредственное присутствие (дежурство) на месте 24 часа в сутки 7 дней в неделю опытных системных администраторов. Но даже в некоммерческой среде вроде университетского факультета в наличии серьезной неполадки в системе на протяжении всех выходных нет ничего хорошего.

Даже если бюджет не позволяет явно обеспечить круглосуточное дежурство, нужно быть готовым к вызову любых администраторов, которые окажутся доступными в позднее время суток или в выходной день. Пейджер или другая система уведомления часто являются "довольно хорошим" средством. Проверьте, чтобы пользователи могли получать доступ к этому средству каким-нибудь простым и хорошо известным способом; например, создайте адрес-псевдоним support-pager ("сигнал в службу поддержки") и

[3] Если файловый сервер выйдет из строя надолго, рассерженные пользователи могут начать настаивать на том, чтобы им разрешили использовать для хранения файлов свои собственные жесткие диски, а не файловый сервер (по крайней мере до тех пор, пока они не столкнутся с отказом локального диска и отсутствием резервной копии файлов).

сделайте так, чтобы все поступающие на этот адрес сообщения автоматически и немедленно перенаправлялись на адрес функционирующего пейджера.

Документирование зависимостей

Для того чтобы делать точные предположения о доступности или периоде безотказной работы, нужно знать не только о своих сильных и слабых местах (степени надежности устанавливаемого оборудования включительно), но и о зависимостях IT-систем от другого оборудования, программного обеспечения и персонала. Ниже перечислены примеры.

- Питание: независимые источники и схемы питания, защита от перепадов напряжения, резервные системы питания, такие как генераторы и ИБП (источники бесперебойного питания), разводка кабелей электропитания в здании, карты питания определенных компонентов оборудования.
- Сеть: проводка в здании, резервные линии связи, служба работы с клиентами для Internet-провайдеров, топология сети, контактная информация для других отделов в организации, выполняющих свою собственную роль в сетевом управлении.
- Оборудование: системы высокой готовности и процедуры для их использования, горячие/холодные резервы, запасные детали, договора на техническое обслуживание оборудования.

Переналадка или списывание старого оборудования

Для того чтобы поддерживать свою инфраструктуру, нужно покупать новые компьютеры, переналаживать устаревшие и выкидывать совсем старые. О закупке оборудования мы рассказывали в разделе, посвященном администрированию, но избавление от старых "любимцев", похоже, чаще всего считается больше обязанностью сотрудников операционного отдела, чем администраторов.

Каждый раз, когда производительность оборудования увеличивается, программное обеспечение тянет ее обратно вниз, обычно тем, что становится более объемным и более сложным. Например, версии Windows 95 и 98 прекрасно функционировали при 32 Мбайт ОЗУ, Windows 2000 для нормальной работы требовала уже, по крайней мере, 128 Мбайт, а ноутбуку, на котором установлена Windows XP, сейчас похоже необходимо уже более 256 Мбайт только для запуска диспетчера задач. Конечно, такое увеличение объема является проблемой не только Windows: современным настольным средам Linux тоже требуются "тонны" памяти.

Поскольку пользователям и руководству (менеджерам) часто не очень хочется обновлять устаревшее оборудование, вы иногда должны брать инициативу в свои руки. Финансовая информация является самым убедительным доказательством. Если вам удастся на бумаге продемонстрировать, что стоимость обслуживания старого оборудования превышает количество средств, требуемых на его замену, многие возражения по поводу обновления отпадут сразу. Иногда гетерогенное оборудование также может быть полезно заменить хотя бы для того, чтобы сэкономить время и усилия, требующиеся на поддержание различных версий ОС и ПО на уровне современных требований.

Недорогое оборудование Intel/PC является стандартной архитектурной базой настольных систем, особенно теперь, когда Apple поставляется на оборудовании Intel. Распространение ПК за последние годы привело к тому, что сегодня дороже обходится ПО и его поддержка, а не оборудование.

Облегчить пользователям переход со старых систем на новые можно, оставив старые системы доступными. Не отключайте старые системы, а просто сократите уровень поддержки, оказываемой по ним отделом системных администраторов. Перестаньте проводить техосмотр старых компьютеров и позвольте им потихоньку доработать свое.

Более старые машины также можно оставить и использовать для выполнения каких-то специальных задач, которые не требуют слишком большого количества аппаратных ресурсов (например, для печати, DHCP или сервера системы лицензирования). Главное — составить план по ремонту или замене этих компьютеров или по переносу установленных на них служб на другой сервер на случай появления проблем.

Университеты часто получают "в подарок" от крупных компаний, которые хотят уменьшить свои налоговые выплаты, старые компьютерные устройства. В большинстве случаев им лучше ответить как-нибудь так: "Нет, спасибо, нам не нужны ни две тысячи девятидорожечных лент, ни стойки для их хранения". Несколько лет назад одному университету в Будапеште предложили мэйнфрейм производства IBM. Вместо того чтобы отказаться от него и купить быстродействующие персональные компьютеры, они согласились и взяли его: доставка и прокладка электропроводки в первый же год обошлась им в сумму, равную всему бюджету. На обслуживание системы в последующие годы тоже уходили все средства. Вряд ли оно того стоило. С другой стороны, многим университетам удается налаживать неплохие отношения с местными компьютерными фирмами и получать от них бесплатно немало ценного оборудования, а именно моделей прошлого или позапрошлого года выпуска. Всегда подсчитывайте общую стоимость владения оборудованием и не бойтесь говорить "нет".

Если вы работаете в компании, и у вас появилось лишнее оборудование, подумайте о том, чтобы подарить его какой-нибудь местной школе или колледжу: у них часто не хватает бюджетных средств на обновление компьютерных классов и потому они могут очень обрадоваться вашему подарку.

Совсем старые компьютеры, которые не подходят даже для благотворительных целей, должны как следует утилизироваться: они содержат некоторые опасные материалы (например, свинец) и поэтому выкидывать их просто в мусорный бак нельзя. Если в организации нет технологического отдела, отвечающего за утилизацию старого оборудования, возможно старые компьютеры придется отнести в один из местных центров по утилизации, чтобы их там уничтожили. Текущая цена утилизации в США составляет 10 долларов за один компьютер.

30.7. Работа отдела поддержки

Задачей отдела поддержки является обслуживание людей, которые пользуются и зависят от установленных в организации систем. Помимо предоставления справочных сведений и круглосуточной поддержки по телефону, этот отдел также может проводить консультации для клиентов и организовывать семинары по конкретным темам. В идеале, эти дополнительные услуги должны увеличивать способность пользователей справляться со всем самостоятельно и, следовательно, снижать количество обращений в службу поддержки.

Доступность

Хорошая IT-поддержка означает, что квалифицированные специалисты доступны и готовы прийти на помощь всегда, когда они нужны клиенту.

Многие проблемы являются незначительными и могут безопасно дожидаться своей очереди. Некоторые же проблемы приводят к остановке работы и требуют немедленного решения. Автоматические ответы от системы отслеживания запросов и записанные на автоответчик сообщения о часах работы вызывают только раздражение. Нужно позаботиться о том, чтобы пользователи всегда могли получить экстренную помощь, если вдруг у них возникнет такая необходимость. Мобильного телефона, переходящего от одного системного администратора другому в нерабочие часы, обычно вполне хватает.

К сожалению, качественная работа отдела поддержки обычно порождает зависимость. У пользователей может очень быстро войти в привычку обращаться в службу поддержки даже по тем вопросам, которым не следует. Если заметите, что кто-то пользуется системой поддержки для получения ответов, которые мог бы найти и на страницах руководства или в Google, начните отвечать на его вопросы цитированием соответствующей `man`-страницы или URL-адреса. Эта тактика прекрасно работает со студентами, однако менее эффективна в отношении руководства.

Объем услуг

Нужно четко определить, какие услуги будет оказывать группа поддержки; если этого не сделать, ожидания пользователей не будут соответствовать действительности. Ниже перечислены некоторые моменты, которые следует продумать:

- время отклика;
- обслуживание в выходные дни и во внеурочное время;
- вызовы на дом (обслуживание компьютеров, установленных у пользователей дома);
- обслуживание специализированного (уникального в своем роде) оборудования;
- обслуживание устаревшего оборудования;
- поддерживаемые операционные системы;
- стандартные конфигурации;
- срок хранения резервных лент;
- специальное программное обеспечение;
- выполнение обязанностей уборщицы (протирание экранов и клавиатур).

В Linux каждый пользователь имеет доступ к исходному коду системы и может легко собрать специальное ядро для свого собственного компьютера. Это следует учесть в корпоративной стратегии и, возможно, попробовать стандартизировать несколько конкретных конфигураций ядра. Если этого не сделать, расчет на то, что обслуживание будет простым, и будет увеличиваться в объеме только по мере разрастания организации, не оправдается. Попросите своих “продвинутых” пользователей сообщить, какие настройки ядра им необходимы для работы. Включите эти настройки в стандартную конфигурацию, и тогда вам не придется заниматься сопровождением множества различных конфигураций ядра.

Пользователи должны быть знакомы не только с перечнем оказываемых услуг, но и с используемой при ответах на запросы схемой приоритетов. Конечно, в любой схеме приоритетов бывают исключения, но нужно попробовать разработать такую схему, которая будет охватывать большинство ситуаций с нулевым или минимальным количеством исключений. Ниже перечислены некоторые из связанных с приоритетами факторов:

- уровень обслуживания, который оплатил или оговорил в контракте клиент;
- количество задействованных пользователей;
- степень важности задействованных пользователей (узнать ее — задача не из легких, но главное не вызвать раздражения у человека, которые подписывает тебе чеки об оплате);
- привередливость задействованных пользователей;
- важность срока выполнения работы (это задание на ночь, предложение поучаствовать в получении гранта на проведение исследований или контракт).

Хотя на расстановку приоритетов будут влиять все эти факторы, мы рекомендуем пользоваться каким-нибудь простым набором правил с разумными поправками на исключительные ситуации. Мы в основном пользуемся следующими приоритетами:

- много людей не могут работать;
- один человек не может работать;
- запросы на улучшение.

Если сразу два или более запросов имеют наивысший приоритет и не могут быть обработаны параллельно, мы принимаем решение, исходя из того, какая из проблем причиняет наибольшие неудобства (например, неработающая служба электронной почты делает недовольными практически всех, в то время как временная недоступность какой-нибудь Web-службы может мешать работе всего лишь нескольких человек). Запросы с низкими приоритетами обычно обрабатываются в порядке поступления.

Поведение сотрудников отдела поддержки можно отточить. Например, мы обычно задаем им вопросы типа: “Как вы поступите, если в то время, когда вы будете работать над какой-нибудь имеющей наивысший приоритет проблемой, к вам в кабинет зайдет возомнивший себя большим начальником пользователь и потребует отремонтировать его принтер, потому что ему нужно распечатать какие-то слайды, которые он собирается продемонстрировать на встрече; такси в аэропорт отъезжает через 15 минут и, конечно, из IT-персонала все кроме вас ушли на обед?”. Любой разумный сотрудник отдела поддержки поймет (по крайней мере, после того, как на секунду задумается), что спорить с этим человеком, которой итак уже со всех сторон не успевает, не имеет никакого смысла. Чтобы обеспечить себе алиби, не помешает спросить, ничего ли, что сломанный файловый сервер останется без присмотра? Если последует утвердительный ответ, тогда разумнее всего будет бросить все и сделать так, чтобы принтер распечатал необходимые слайды.

Затем мы спрашиваем: “А как вы поступите, когда этот же человек снова вовлечет вас в похожую ситуацию несколько недель спустя?”. Опять-таки, спорить — бессмысленно, но на этот раз пожалуй будет неплохо назначить ему встречу после того, как он вернется, дабы обсудить его потребности в поддержке. Возможно, ему нужен отдельный специалист, который будет обслуживать только его, и если он согласен оплачивать услуги этого специалиста, это только плюс. Главное не пытаться с ним спорить, когда он волнуется по поводу того, что может опоздать на свой самолет.

Пользователи обычно думают, что все их важные данные хранятся на резервных лентах, которые будут лежать в архиве вечно. Но ленты не бессмертны: магнитные носители имеют ограниченный срок службы, после которого считать с них данных становится трудно. (При желании, чтобы данные хранились как можно дольше, их следует периодически перезаписывать на какой-нибудь новый носитель.) Резервные ленты также

могут служить доказательством в суде, из-за чего организация может не хотеть, чтобы ее старые данные были доступны вечно. Наилучшее решение — пообщаться с людьми, которые отвечают за принятие подобных решений, и составить вместе с ними письменный договор, оговаривающий сроки хранения резервных копий данных, их количество (обязательное? желательное? недопустимое?) и то, должны ли они храниться в разных местах.

Эту информацию следует делать доступной для пользователей, чтобы они имели реалистичное представление о резервных копиях своих данных и также могли принять свои собственные меры, если чувствуют, что их данные нуждаются в более надежной защите.

Перечень навыков

Ничто не раздражает опытного пользователя больше, чем человек из команды поддержки, который спрашивает: “А вы подключили кабель питания?” — пока на самом деле просто пытается отыскать его в базе данных клиентов. С другой стороны, заставлять самого опытного администратора объяснять пользователю-новичку, как отыскать клавишу <Delete> в какой-нибудь системе обработки текстов — это тоже просто неэффективное использование ресурсов.

У нас каждый сотрудник обязательно должен отрабатывать в службе поддержки определенное количество часов в неделю[4]. Сотрудники отдела поддержки включают это время в свой еженедельный план работ и затем вызывают человека, когда для него появляется задание. Задания распределяются в соответствии с навыками, требующимися для устранения проблемы, и количеством времени, оставшимся у каждого в еженедельном резерве отдела поддержки.

Для того чтобы эта схема работала успешно, необходим сбалансированный перечень навыков сотрудников. На протяжении определенного срока каждый из сотрудников должен отработать свое количество часов в службе поддержки. В целом, сотрудник, имеющий множество записей в списке навыков, является более “ценным”. Однако в сотруднике с меньшим количеством навыков тоже нет ничего плохого, если для него всегда хватает работы.

Точный список навыков помогает определить, достаточно ли в организации работает сотрудников с определенными навыками, чтобы можно было не волноваться, если кто-то из них уйдет на больничный или в отпуск. Список навыков можно составлять постепенно, по мере появления проблем и их решения различными сотрудниками, включая в него наименование проблемы, имя занимавшегося ею сотрудника и продемонстрированный им при ее решении уровень мастерства.

Навыки следует описывать с определенной степенью точности, т.е. и не слишком конкретно, и не слишком обще. Приведенный ниже в качестве примера список демонстрирует наиболее оптимальный вариант перечисления навыков.

- Создание пользователей, удаление пользователей, установка паролей, изменение квот.
- Создание учетных записей CVS или SVN.
- Интегрирование новых драйверов оборудования в RIS (Windows).

[4] По умолчанию каждый системный администратор должен отрабатывать в отделе поддержки 25% всего своего рабочего времени. Когда проект требует постоянного участия того или иного системного администратора, мы отклоняемся от стандартной схемы и освобождаем его от работы в службе поддержки на время этого проекта.

- Упаковка Windows-приложения в формате MSI.
- Создание и установка пакетов программ в Linux.
- Анализ журнальных файлов.
- Устранение неполадок, связанных с почтовым сервером.
- Отладка проблем, связанных с печатью.
- Устранение общих неполадок с оборудованием.
- Добавление записей в службу DNS.
- Решение вопросов, касающихся лицензий на программное обеспечение.
- Умение отвечать на вопросы, касающиеся безопасности в системах Windows.
- Умение отвечать на вопросы, касающиеся безопасности в системах UNIX.
- Разрешение запросов по Samba.
- Конфигурирование сервера DHCP.
- Настройка сервера LDAP.
- Добавление и удаление Web-сайтов (конфигурирование сервера Apache).

Управление временем

Системное администрирование подразумевает такое количество переключений с одного контекста на другой за день, которое на других работах не происходит и за целый год, и в основном с этим хаосом приходится справляться именно сотрудникам отдела поддержки пользователей. Каждый администратор должен обладать навыками управления временем. Без них он не сможет справляться со своими повседневными обязанностями и, следовательно, станет раздраженным или вообще впадет в депрессию. (Если же он уже постоянно находится в раздраженном или депрессивном состоянии, ему станет даже еще хуже.)

Системные администраторы “изнашиваются” очень быстро. Многих из них хватает всего лишь на несколько лет. Никто не хочет, чтобы его все время вызывали и постоянно отчитывали. Умение эффективно распределять время и успевать обслуживать пользователей так, чтобы они оставались довольными, является очень выигрышным качеством.

В своей недавно опубликованной книге *Time Management for System Administrators* автор, Том Лимонцелли (Tom Limoncelli), предлагает несколько способов избежать таких проблем. Полный перечень источников приводится в конце этой главы.

30.8. Документация

Точно так же как большинство людей соглашается с благотворным влиянием на организм физических упражнений и свежих овощей, каждый ценит хорошую документацию и догадывается о ее важности. К сожалению, это вовсе не обязательно означает, что она будет составляться и обновляться ими без напоминаний.

Действительно, почему мы должны волноваться о документации?

- Документация снижает вероятность одиночных сбоев. Замечательно, когда есть программы, которые позволяют мгновенно разворачивать рабочие станции и распространять заплаты и обновления при помощи одной единственной команды, но они становятся практически бесполезными, если по ним нет никакой документа-

ции, а умеющий с ними обращаться специалист ушел в отпуск или вообще уволился.

- Документация способствует воспроизводимости. Если правила и процедуры не хранятся в документах организации, ими вряд ли будут пользоваться регулярно. Когда администраторы не могут отыскать информацию о том, как следует поступать в том или ином случае, им приходится импровизировать.
- Документация экономит время. При написании документации вовсе не кажется, что этим экономиться время, но потратив несколько дней на решение проблемы, которая когда-то раньше решалась, но как именно — уже никто не помнит, большинство администраторов соглашается, что написание документации того стоит.
- И, наконец, самое главное: документация делает систему более понятной и позволяет вносить последующие изменения совместимым с устройством системы образом. Когда изменения вносятся на основе только частичных знаний, они часто не совсем вписываются в имеющуюся архитектуру. Со временем степень несоответствия увеличивается, и система даже работающим с ней администраторам начинается казаться беспорядочной коллекцией компонентов. В конечном итоге часто возникает желание удалить все компоненты и начать все заново.

Локальная документация удобна во многих ситуациях. Доводилось ли вам заходить в машинный зал для выполнения перезагрузки одного сервера и сталкиваться с целыми рядами стоек с похожей и в то же время разной аппаратурой, никак не подписанной? Или приходилось ли вам устанавливать устройство, с которым вы уже имели дело когда-то давно, но сейчас помните только то, что настраивать его было очень трудно?

Локальная документация должна храниться в четко установленном месте. В зависимости от объема выполняемых операций, это может быть либо каталог на файловом сервере, смонтированный на всех компьютерах, либо даже домашний каталог, принадлежащий какой-то специальной учетной записи в системе.

Стандартизированная документация

Как показывает наш опыт, самые простым и эффективным способом ведения документации является стандартизация небольших, простых документов. Вместо того чтобы писать целое руководство по системному управлению для своей организации, лучше подготовить ряд одностраничных документов, посвятив каждый из них какой-нибудь одной конкретной теме. Начните с общей картины, а затем разбейте ее на отдельные части, содержащие дополнительную информацию. При необходимости добавить где-то больше деталей, напишите дополнительный документ и перечислите в нем шаги, которые являются особо сложными или запутанными.

Такой подход имеет несколько преимуществ.

- Начальника наверняка интересует только общая настройка компьютерной среды в организации. Этого будет вполне достаточно, чтобы ответить на все вопросы как начальства, так и менеджеров. В глубокие подробности лучше не вдаваться, а то начальник еще решит разобраться во всех них.
- То же верно и для клиентов.
- Любому новому сотруднику или человеку, переведенному на новую должность, необходимо быстро ознакомиться с инфраструктурой, чтобы нормально включиться в производственный процесс. “Загружать” таких людей деталями не имеет никакого смысла, это только замедлит их адаптацию на новом месте.

- Намного эффективнее и быстрее выбрать один подходящий короткий документ, чем искать нужную информацию в длинном руководстве.
- Поддерживать документацию в "текущем" состоянии намного проще, когда обновлять можно по одной странице.

Последний пункт является особенно важным. Обновление документации — это непростая задача; ее часто откладывают первой, когда не хватает времени. Мы обнаружили несколько подходов, которые позволяют поддерживать документацию в порядке.

Во-первых, нужно установить правило, что документация должна быть краткой, важной и вовсе необязательно "ошлифованной". Главное передать суть. Ничто так не отбивает желание составлять документацию, как осознание того, что придется писать целую "диссертацию" на тему проектирования. Предъявление высоких требований к составлению документации приведет только к тому, что никто вообще не будет ее составлять.

Во-вторых, нужно внедрить документацию в процессы. Комментарии в конфигурационных файлах являются чуть ли не самой лучшей документацией из всех. Они всегда там, где они нужны, и их сопровождение (обновление) занимает совсем немного времени. Во многие стандартные конфигурационные файлы разрешается добавлять комментарии и даже в те из них, которые не очень "дружат" с комментариями, часто все равно можно вставить какую-нибудь дополнительную информацию. Например, стандартное содержимое поля GECOS в файле `passwd` (представляющее собой название компании, номер телефона и т.д.) часто оказывается не самой полезной информацией, которая могла бы там находиться. Не стесняйтесь определять свои собственные условные обозначения в пределах своей организации. (Но в этом конкретном случае не забывайте, что пользователи могут изменять содержимое этого поля, поэтому доверять хранящейся там информации можно будет не всегда.)

Создаваемые локально средства могут требовать документацию в качестве части их стандартной конфигурационной информации. Например, инструменту, выполняющему настройку нового компьютера, может быть нужна информация о владельце компьютера, размещении, уровне поддержки и оплате, даже если эти факты не влияют непосредственно на конфигурацию программных средств компьютера.

Документация не должна приводить к информационной избыточности. Например, если в организации имеется главный конфигурационный файл, в котором перечислены все компьютеры и их IP-адреса, не должно быть больше никакого другого места, где эта же информация могла бы быть обновлена вручную. Это чревато не только тратой дополнительного времени на обновление многочисленных источников, но и неизбежным появлением в них через какое-то время определенных несоответствий. При возникновении необходимости использовать эту информацию в другом контексте или другом конфигурационном файле, следует просто написать сценарий, который будет извлекать ее из главного конфигурационного файла (или обновлять ее там). Если устранить все избыточные источники не представляется возможным, нужно хотя бы определиться с тем, какой из них будет главным, а также написать программу для перехвата несоответствий и сделать так, чтобы она запускалась регулярно, например, из `cron`.

Маркировка оборудования

Некоторая документация выглядит наиболее подходяще, когда написана на бумажке и приклеена к устройству. Например, от перечня действий в случае отказа всей системы или сети мало толку, если он хранится на сломавшемся или недоступном компьютере.

Нужно, чтобы каждый компьютер можно было идентифицировать без включения и прохождения регистрации, потому что эти операции не всегда будут возможны. Промаркируйте все рабочие станции каким-нибудь уникальным образом (например, путем указания имени хоста и IP-адреса) и наклейте на них бумажки с контактной информацией службы поддержки.

В серверном зале все системы и их внешние устройства должны быть промаркированы по имени хоста (полностью уточненным, если там находятся машины из разных доменов). Эти маркировочные надписи лучше наклеить с обеих сторон компьютеров (особенно тех, что установлены в узких стойках), чтобы при необходимости отключить и снова включить какой-нибудь компьютер, можно было быстро отыскать отключающий именно его питание выключатель.

При наличии множества систем разных типов может быть полезно добавить дополнительную информацию, такую как сведения об архитектуре, инструкции по загрузке, специальные последовательности нажатия клавиш, ссылки на дополнительную документацию, телефон "горячей линии" производителя или номер телефона человека, который за все это отвечает. Записывание последовательностей нажатия клавиш может показаться немного глупым занятием, но серверы часто подключены к устаревающему терминалу или консоли, а не к специальному монитору.

Копию информации со всех наклеек следует обязательно сохранить также и в центральном хранилище записей или инвентаризационных данных. Она пригодится, если вы решите управлять своими компьютерами через TCP/IP-соединение с сервером консоли вместо того, чтобы проводить рабочий день в шумном машинном зале.

Имя хоста также следует записать и на всех остальных устройствах, которые ассоциируются с каждым компьютером, таких как дисководы, модемы, принтеры, лентопротяжные устройства и т.д. Если несколько идентичных внешних подсистем подключены к одному и тому же хосту, нужно убедиться в том, что надписи являются уникальными[5]. У нас подписаны даже кабели SCSI и Fibre Channel (и, конечно же, разные порты на сервере), так что нам на самом деле точно известно, к какому интерфейсу подключено каждое устройство.

Если хост является важным компонентом (например, основным сервером или ключевым маршрутизатором), не помешает также указать местонахождение его выключателя. Если для загрузки требуется дискета или флэш-карта, обязательно следует указать, где она находится. У основных файловых серверов должна быть информация об именах дисковых устройств, таблицах разделов, точках монтирования и местонахождении готовых к использованию запасных суперблоков. Запишите эту информацию на самих дисковых подсистемах и храните ее в хорошо известном месте, например в машинном зале.

Лентопротяжные устройства должны быть снабжены пометками с перечнем их файлов и командами, необходимыми для получения к ним доступа. Также не помешает перечислить тип лент, которые используют эти устройства, ближайшее место, где их можно купить, и даже приблизительную цену.

Принтеры следует снабдить пометками с их именами, краткими инструкциями по печати и именами хостов, от которых они зависят. Принтеры часто поставляются с сетевыми интерфейсами и являются полноправными компонентами сети, но в вопросах спулинга и конфигурации они все равно зависят от хоста Linux.

[5] Один из наших сотрудников недавно заменил запасную батарею кэша записи не в том RAID-массиве. К счастью, эта ошибка закончилась лишь незначительным снижением производительности. Но представьте, что произойдет, если вы замените не тот диск в RAID5-массиве, не заметите этого, а потом у вас откажет второй диск.

Сетевая проводка должна быть тщательно документирована. Подпишите все кабели, обозначьте коммутационные панели и сетевые розетки и промаркируйте сетевые устройства. Всегда делайте так, чтобы специалисту по проводке было удобно обновлять документацию; повесьте на стенку в коммутационном шкафу ящик с карандашом и формами для того, чтобы он мог всегда быстро записать, что такой-то кабель был отключен от одного устройства и подключен к другому. И не забывайте позже переносить эти данные в электронное хранилище.

Дело усложняется еще больше, если сетевые устройства (такие как маршрутизаторы и коммутаторы) могут быть переконфигурированы по сети. Хотя это и позволяет управлять компьютерами в разных подсетях, не выходя из своего удобного кабинета, документация становится даже еще более важной. В таком случае следует соблюдать еще большую осторожность, потому что можно быстрее и безнадежнее испортить гораздо большую часть инфраструктуры.

Документация для пользователей

Неплохой идеей является подготовить печатный документ, который можно будет выдавать новым пользователям. В этом документе следует перечислить местные "обычаи", правила уведомления о проблемах, имена и местонахождение принтеров, графики резервного копирования и простоев и т.д. Такой документ может сэкономить системным администраторам и пользователям огромное количество времени. Эту информацию также следует сделать доступной в Web. Печатный документ дает большую уверенность в том, что новые пользователи его все-таки прочитают, зато на Web-страницу удобно ссылаться в случае возникновения вопросов. Поэтому лучше сделать и печатную, и Web-версию документа и затем не забывать регулярно их обновлять. Ничто не раздражает так, как устарелая электронная документация или раздел часто задаваемых вопросов (FAQ).

Помимо документа с информацией о местной компьютерной среде, можно также подготовить какой-нибудь вводный материал по Linux. Такой материал очень важен в университетской среде, где пользователи постоянно меняются и зачастую плохо знакомы с Linux. Мы предоставляем нашим пользователем одностраничные "шпаргалки", в которых перечислены наиболее часто используемые в нашей среде команды и приложения.

30.9. Системы отслеживания запросов и уведомления о проблемах

При наличии большого штата сотрудников не помешает упорядочить рабочий поток при помощи системы отслеживания запросов. Даже если в IT-отделе работает всего лишь один человек, такая система может очень упростить отслеживание всех нуждающихся в решении проблем и запросов. Системы отслеживания запросов также являются замечательным источником хронологической информации о проделанной командой системных администраторов работе и удобным средством контроля внутренних планов работ и проектов. Руководители всегда хотят знать, чем занимаются системные администраторы: почти все такие системы умеют генерировать отчеты, демонстрирующие рабочую нагрузку, ресурсы и острую потребность в дополнительных сотрудниках и большем бюджете.

Эти системы называют по-разному: кто-то называет их "системами отслеживания запросов" (request tracking systems), кто-то — "диагностическими системами" (trouble ticket systems), а кто-то — "системами отслеживания неполадок" (bug trackers). Но все

это одни и те же системы. О распространяемых с открытым исходным кодом реализациях этих систем можно подробнее узнать на сайте `linas.org/linux/pm.html`.

Общие функции диагностических систем

Диагностическая система принимает запросы, поступающие через различные интерфейсы (среди которых наиболее распространенными являются электронная почта, Web-формы и командные строки), и следит за ними от самого момента их поступления до того момента, когда они будут решены. Руководители могут назначать обработку запросов и сообщений о неполадках как группам, так и отдельным системных администраторам. Системные администраторы, в свою очередь, могут запрашивать у диагностической системы информацию об очереди ожидающих ответа запросов и даже по собственной инициативе обрабатывать некоторые из них. Пользователям диагностическая система позволяет просматривать информацию о состоянии своего запроса и том, кто им занимается, а менеджерам — извлекать высокоуровневые данные, такие как данные о:

- количестве обрабатываемых запросов или сообщений о неисправностях;
- количестве времени, затрачиваемом в среднем на обработку сообщений о неисправностях;
- производительности системных администраторов;
- количестве необработанных запросов (в процентах);
- распределении рабочей нагрузки по времени для обработки запросов.

Данные запросов, сохраняемые в диагностической системе, превращаются в данные о возникавших в имеющейся IT-инфраструктуре проблемах и принятых решениях. А если по этим данным еще и можно быстро выполнять поиск, они становятся просто бесценным ресурсом для системных администраторов.

Сообщения с описанием решения проблемы могут рассылаться начинающим или обучающимся системным администраторам, вставляться в систему часто задаваемых вопросов или просто регистрироваться (заноситься в журнал). Новым сотрудникам может быть полезно ознакомление с закрытыми отчетами о неполадках (т.е. обработанными запросами), потому что эти отчеты могут содержать не только техническую информацию, но и примеры тона и стиля поведения с клиентами.

Как и все документы, хронологические данные диагностической системы потенциально могут быть использованы против организации в суде. Поэтому четко следуйте установленным юридическим отделом правилам касательно срока хранения документов.

Большинство систем отслеживания запросов автоматически подтверждают принятие новых запросов и присваивают им номер, при помощи которого пользователи могут отслеживать свои запросы и запрашивать информацию об их состоянии. Отправляемое автоматически ответное сообщение должно давать ясно понять, что это лишь подтверждение получения запроса. За ним как можно быстрее должно следовать сообщение от реального человека с объяснениями и планом решения указанной в запросе проблемы.

Отношение пользователей к диагностическим системам

Быстрое получение ответа от реального человека является критическим показателем для пользователей, даже если в личном ответе содержится не больше информации, чем в автоматическом. В большинстве случаев гораздо важнее уведомить отправившего запрос пользователя о том, что его запрос был просмотрен реальным специалистом, чем не-

медленно устранить проблему. Пользователи понимают, что администраторы получают много запросов, и готовы дожидаться того времени, когда им смогут уделить достаточно времени. Но они не любят, когда их игнорируют.

В нашей диагностической системе используется почтовый псевдоним “support”. Одно время нас забрасывали неполными или непонятными сообщениями о неисправностях. Поэтому мы написали сценарий, который задавал пользователю конкретные вопросы вроде:

- На каком компьютере возникла проблема?
- Повторяется ли она?
- Насколько важно устранить эту проблему немедленно?

Примерно через час пользователи подняли бунт, поэтому уже к концу дня нам пришлось все отменить. Но, похоже, что это все-таки дало свой результат, потому что многие пользователи хоть и злились на сценарий, но все равно прочитали наши вопросы, и качество составляемых им в свободной форме сообщений о неисправностях заметно повысилось.

В другой организации к этой проблеме подошли так: всем пользователям был разослан документ с объяснением того, как какая информация обязательно должна включаться в сообщение о неисправностях, и примерами неправильно составленных запросов. В случае получения некорректно составленного запроса в ответ отправлялось сообщение с извинениями (наподобие: “Извините, но у нас недостаточно информации для того, чтобы...”) и копией этого пояснительного документа. Пользователи быстро разобрались, что к чему.

Согласно нашему опыту, формы подходят только для строго стандартизированных задач, таких как запросы на создание или удаление учетных записей. В таком контексте пользователи понимают, что запрашиваемая информация просто необходима администратору для выполнения его работы. Некоторые задачи, такие как настройка сообщения об отсутствии для пользователя, могут выполняться полностью автоматически после заполнения формы. Другие, такие как создание учетной записи, могут быть автоматизированы частично и требовать авторизации или других действий со стороны системного администратора.

Диагностические системы

В приводящихся ниже таблицах 30.1 и 30.2 кратко перечислены характеристики нескольких хорошо известных диагностических систем. В табл. 30.1 описываются системы с открытым исходным кодом, а в табл. 30.2 — коммерческие системы.

Нам очень нравится Mantis. Первоначально она была разработана для отслеживания неполадок в программном обеспечении видеоигр. Она запускается на Linux, Solaris, Windows, Mac OS и даже OS/2. Она занимает мало места, проста, легко изменяется и конфигурируется. Требует наличия PHP, MySQL и Web-сервера. Но ее самым важным качеством является хорошая документация!

OTRS (Open Ticket Request System — система запроса открытых документов) — тоже неплохая система. В ней доступны интерфейсы как для клиентов, так и для системных администраторов, а также интерфейс для электронной почты. OTRS позволяет настраивать очень много параметров (например, она позволяет по очереди настраивать приветственные сообщения) и умеет даже фиксировать время, затраченное на обработку сообщений о неполадках. Пакеты для всех рассматриваемых нами платформ Linux доступны на Web-сайте OTRS.

Таблица 30.1. Диагностические системы, распространяемые с открытым исходным кодом

Название	Источник входных данных[а]	Язык	База данных[б]	URL-адрес
Mantis	WE	PHP	M	www.mantisbt.org
RT: Request Tracker	W	Perl	M	www.bestpractical.com
Scarab	W	Java	M	scarab.tigris.org
Double Choco Latte	W	PHP	PM	dcl.sourceforge.net
OTRS	WE	Perl	PMOD	www.otrs.org
JitterBug[в]	WE	C	F	www.samba.org/jitterbug
WREQ	WE	Perl	G	math.duke.edu/~yu/wreq

[а] Источники входных данных: W=Web, E=Email=электронная почта.

[б] База данных: M = MySQL, P = PostgreSQL, O = Oracle, D = DB2, F = плоские файлы, G = gdbm.

[в] Больше не поддерживается.

RT: Request Tracker (выпуск 3) имеет очень богатый набор функциональных возможностей, включая поддающиеся очень точной настройке Web-интерфейсы как для пользователей, так и для системных администраторов. Поскольку в состав этой системы входит множество модулей Perl, ее может быть немного трудно устанавливать. Ее интерфейс командной строки тоже является довольно-таки примитивным. В некоторых организациях используют первую версию RT с заплатами, которые устраняют некоторые из ее наиболее раздражающих недостатков, не добавляя сложности последних версий.

В табл. 30.2 перечислено несколько коммерческих программ для управления запросами. Их язык реализации и базы данных здесь не указаны, потому что эта информация в полном объеме доступна на посвященных им Web-сайтах.

Таблица 30.2. Коммерческие диагностические системы

Название	Размер	URL-адрес
Remedy (новое название — BMC)	Огромный	www.bmc.com/remedy
ServiceDesk	Огромный	manageengine.adventnet.com
HEAT	Средний	www.frontrange.com
Track-It!	Средний	www.numarasoftware.com

Некоторые из этих коммерческих систем настолько сложны, что для их сопровождения, настройки и поддержания в рабочем состоянии нужно нанимать отдельного человека или даже двух (для тех, кто не знает, сообщаем, что в данном случае речь идет о системе Remedy и ServiceDesk). Такие системы подходят для организаций с большим штатом IT-сотрудников и являются неудачным выбором при наличии типичной, небольшой команды IT-специалистов, которые итак постоянно завалены работой.

Распределение сообщений о неисправностях

В большой организации, даже в той, где установлена замечательная диагностическая система, все равно остается решить еще одну проблему. Не очень хорошо, когда системным администраторам приходится разрываться между задачей, над которой они сейчас

работают, и очередью запросов, особенно если запросы поступают по электронной почте прямо в их личный почтовый ящик. Мы испробовали два подхода для решения этой проблемы.

В первый раз мы организовали для наших системных администраторов смены по полдня. Каждый во время своей смены должен быть стараться ответить на как можно большее количество поступивших запросов. Проблема этого подхода состояла в том, что не у всех системных администраторов были навыки, необходимые для ответа на все вопросы и устранения всех проблем. Иногда ответы были некорректными из-за того, что заступивший на смену человек был новичком и еще плохо знал клиентов, их среды или предусмотренные в их контракте услуги по поддержке. В результате старшим администраторам все равно приходилось присматривать за всеми остальными, из-за чего они, естественно, не могли полностью сконцентрироваться на своей работе. В конечном итоге качество обслуживания только ухудшилось, и никакой пользы этот подход не принес.

После такого неудачного опыта мы придумали роль "диспетчера", которая ежемесячно переходит от одного старшего администратора к другому. Диспетчер отвечает за проверку диагностической системы на наличие новых записей и распределяет задачи между конкретными сотрудниками. При необходимости диспетчер связывается с пользователями и узнает у них дополнительные сведения, необходимые для определения степени важности запросов. Диспетчер пользуется специальной базой данных о навыках сотрудников, созданной в организации своими силами, для принятия решений о том, у кого в группе поддержке имеются необходимые навыки и время. Кроме того, диспетчер также следит за тем, чтобы запросы обрабатывались своевременно.

30.10. Восстановление после аварий

Работа организации зависит от функционирования IT-среды. Системный администратор отвечает не только за повседневные операции, но и за наличие плана действий на случай возникновения различных экстренных обстоятельств, по крайней мере тех, которые можно предвидеть. Подготовка к таким масштабным проблемам влияет как на общий план работы, так и на способ выполнения повседневных операций.

В этом разделе мы расскажем о различных видах аварий, данных, которые необходимы для того, чтобы аккуратно восстановить систему, и важных пунктах, которые обязательно должны присутствовать в аварийном плане.

Мы рекомендуем написать подробный перечень потенциальных экстренных случаев и аварий, от которых хотелось бы защититься. Все аварии выглядят по-разному, поэтому не исключено, что придется составить несколько разных планов, чтобы охватить весь спектр возможностей. Ниже перечислены несколько наиболее распространенных типов аварий и непредвиденных ситуаций.

- Взлом системы: до 2000 г. 60% попыток взлома предпринималось внутри организации. В 2001 г. из-за большого числа внешних атак этот показатель снизился до 30%. В 2005 г. количество проводимых внутри организаций атак снова увеличилось до 47%[6].
- Проблемы внешнего характера: скачки напряжения и отключение питания, сбои в работе системы охлаждения и вентиляции, наводнения, ураганы, землетрясения, метеориты, террористы или вторжение пришельцев из космоса.

[6] Эти данные были получены от ФБР и Института компьютерной безопасности США.

- Человеческие ошибки: удаление или повреждение файлов и баз данных, потеря конфигурационной информации.
- Неожиданный выход из строя оборудования: отказ сервера, поломка жесткого диска, нарушение работы сети.

Спросите себя, какие преимущества дает пользователям IT-среда? Какие из них являются наиболее важными? Что может лишить пользователей этих преимуществ? Как от этого можно защититься? Правильно сформулировать проблему — это непростая задача, потому очень важно, чтобы в этом принимали участие не только системные администраторы, а все работающие в данной организации сотрудники.

Резервное копирование и хранение информации вне сети

По очевидным причинам процедура резервного копирования является неотъемлемой частью планирования действий на случай аварийных ситуаций; более подробно она описывалась в главе 9. Но создание надежных резервных копий, хранящихся за пределами организации — это только полдела. Насколько быстро можно будет получить доступ к этим резервным копиям? Как придется собирать вместе запасное оборудование, человеческие ресурсы и резервные данные, чтобы заменить сломавшийся сервер?

Мы привыкли пользоваться сетью для общения и получения доступа к документам. Но в случае инцидента информация, хранящаяся в сети, может оказаться недоступной или поврежденной. Поэтому всю контактную информацию и перечни инструкций следует хранить вне сети. То, где находятся последние архивные ленты и какую команду `restore` следует использовать для их восстановления, нужно помнить наизусть, не заглядывая в файл `/etc/dumpdates`.

В случае любой аварии администратору потребуется доступ как к онлайновым, так и к автономным копиям основных данных. Онлайновые копии следует, если возможно, хранить на каком-нибудь независимом компьютере с богатым набором всевозможных утилит и инструментальных средств, ключевыми программами системных администраторов, собственным сервером имен, полной локальной версией файла `/etc/hosts`, отсутствием зависимостей от сетевых файлов, подключенным принтером и т.д. Не стоит использовать для этой цели совсем старую машину, которая просто больше ни для чего другого не годится; компьютер, предназначенный для проведения процедур по восстановлению системы после аварий, должен быть быстрым и иметь много памяти и свободного пространства на диске, чтобы его можно было использовать для выполнения восстановительных и сравнительных операций. На нем должна быть установлены все средства разработки, только тогда на нем можно будет добавлять заплаты и заново компилировать поврежденное программное обеспечение. Также не помешает, если на этом компьютере будут доступны интерфейсы для всех типов дисков, которые используются в данной организации (IDE, SATA, SCSI, FC-AL и т.д.).

Ниже приводится перечень нужных данных, которые следует хранить на резервном компьютере и иметь под рукой в распечатанном виде:

- план действий персонала на случай аварии (должно быть указано, кого и когда вызывать и что говорить);
- номера телефонов обслуживающих организаций и клиентов;
- важнейшие местные номера телефонов (такие как номера телефонов администраторов, полиции, пожарной службы, начальства, агентства по трудоустройству);

- сведения о конфигурации аппаратного и программного обеспечения: версия операционной системы, уровни заплат, таблицы разделов дисков, аппаратные установки, номера прерываний, номера каналов DMA и т.п.[7];
- ленты с резервными копиями и график, по которому они создавались;
- схемы сети;
- серийные номера программного обеспечения, лицензионные данные и пароли;
- контактная информация производителей дисков на случай, если какой-то диск понадобится достать немедленно.

Подбор персонала на случай аварии

Нужно заблаговременно решить вопрос о том, кто будет справляться с ситуацией в случае, если произойдет авария. Следует составить план действий и написать номера телефонов тех, кому надлежит звонить в такой ситуации. Может оказаться, что на роль главного больше подходит какой-нибудь из системных администраторов, а не начальник IT-отдела (начальники IT-отделов обычно не подходят для этой роли).

Ответственным лицом в случае аварии должен быть кто-то, кто пользуется авторитетом и не боится принимать трудные решения при наличии минимального количества информации (наподобие решения отключить от сети весь отдел целиком). Способность принимать такие решения, уведомлять о них понятным образом и фактически руководить персоналом во время кризиса, пожалуй, является важнее теоретических знаний о системном и сетевом управлении. Мы пользуемся небольшими ламинированными карточками, на которых микроскопическим шрифтом напечатаны все важные имена и номера телефонов: они очень удобны, потому что легко умещаются в бумажнике.

При составлении плана аварийных мероприятий обычно предполагается, что административный персонал будет на месте, когда случится авария, и сумеет справиться с ситуацией. К сожалению, люди иногда болеют, уходят на повышение квалификации, уезжают в отпуск, а в тяжелые времена вообще могут даже становиться крайне недружелюбными. Поэтому стоит заранее продумать, где в случае чего можно будет быстро найти дополнительную помощь. (Если система не слишком устойчива, а персонал неопытен, то недостаточное количество администраторов уже само по себе является аварийной ситуацией.)

Одним из решений может быть договоренность с какой-нибудь местной консультационной компанией или университетом, где всегда есть талантливые и готовые помочь администраторы. Конечно, если у них когда-нибудь возникнут проблемы, тогда вы тоже должны будете оказать им необходимую помощь. Но самое главное — это не работать на пределе: наймите достаточное количество администраторов и не требуйте, чтобы они работали по 12 часов в сутки.

Электропитание и кондиционирование

План аварийных мероприятий лучше проверить до того, как он понадобится. Генераторы и источники бесперебойного питания (ИБП) следует проверять раз в месяц или раз в квартал, в зависимости от того, на какую степень риска готово согласиться

[7] Сетевые поисковые и инвентаризационные средства могут компилировать многие из этих данных автоматически. Двумя наиболее заслуживающими внимания кандидатами на эту роль являются H-Inventory (`sourceforge.net/projects/h-inventory`) и LANsurveyor (`www.neon.com`).

руководство. Нужно убедиться в том, что все важные устройства подключены к ИБП, их батареи в порядке и механизм включения питания работает. Чтобы проверить какой-нибудь отдельный ИБП, нужно просто вынуть вилку из розетки, а чтобы проверить все ИБП, т.е. то, все ли важное оборудование к ним подключено, возможно, придется отключить питание во всем здании или комнате. В любом случае следует обязательно знать обо всех зависимостях и слабых местах своей системы электропитания.

ИБП тоже нуждаются в обслуживании. Эта задача может и не входит в обязанности системного администратора, но именно он должен следить за тем, чтобы она выполнялась.

Чаще всего электричество отключается ненадолго, но на всякий случай батареи должны обеспечивать два часа работы, чтобы было время правильно выключить технику. Некоторые ИБП оборудованы последовательными портами или интерфейсом Ethernet, которые можно использовать для корректного выключения не являющихся критически важными компьютеров через пять минут (этот параметр настраивается) после отключения питания.

Периоды отключения электропитания можно использовать для выполнения любых пятиминутных процедур обновления, которые уже были протестированы, но еще не проводились. Так или иначе, но работать невозможно, так что людям будет все равно. В некоторых организациях пользователи спокойнее воспринимают дополнительную пятиминутную задержку после отключения электричества, чем пятиминутное плановое отключение системы, о котором их оповестили за неделю. Если есть старые компьютеры, которыми, вроде бы никто уже не пользуется, можно оставить их отключенными до тех пор, пока кто-нибудь не пожалуется на их отсутствие. Иногда отсутствие таких компьютеров остается незамеченным в течение нескольких недель, а иногда о них вообще никто так никогда и не вспоминает.

Системы охлаждения часто оборудованы датчиками температуры со средствами оповещения о ее повышении. Задайте такую верхнюю границу температуры, чтобы после сигнала у вас оставалось время выключить технику, прежде чем она перегреется и выйдет из строя; например, мы пользуемся отметкой в 76 градусов по Фаренгейту (что соответствует 24 градусам по Цельсию) вместо 90 (32 градуса по Цельсию), но живем в горах в сорока пяти минутах езды (летом; зимой на то, чтобы добраться до места может уходить и гораздо больше времени). В машинном зале лучше еще держать несколько обычных или работающих от батареи термометров, ведь при отключении электропитания все электронные индикаторы станут бесполезными.

О проблемах, связанных с влиянием внешних факторов, более подробно рассказывалось в разделе 24.7.

В одной крупной правительственной лаборатории США недавно построили новый машинный зал и установили в нем для работы со сложными научными моделями кластер из 256 процессоров Alpha. Все оборудование подключили к источникам бесперебойного питания и снабдили всеми самыми современными средствами обслуживания. К сожалению, случившийся через некоторое время перебой в питании вывел всю эту систему из строя на целых четыре часа. Почему? Оказалось, что компьютер, контролировавший работу системы вентиляции и кондиционирования воздуха, не был подключен к ИБП. Он отключился, и это естественно привело к отключению системы кондиционирования. Так что тестируйте свои системы как можно тщательнее.

Если какая-то часть оборудования устанавливается в удаленном месте, попросите отвечающие за обслуживание этого помещения службы показать резервные средства

питания, прежде чем подписывать с ними договор. Удостоверьтесь в том, что генератор находится в рабочем состоянии и проверяется регулярно. Попросите разрешения присутствовать при следующей проверке генератора; независимо от того, разрешат вам это сделать или нет, вам, скорее всего, удастся таким путем получить очень полезную информацию.

Сетевая избыточность

Провайдеры Internet часто поглощаются более крупными компаниями. Это сводит на нет их тщательно продуманные планы по поддержанию избыточных соединений с Internet. Поглощаемые Internet-провайдеры часто принимают линии связи, принадлежавшие независимым компаниям. В результате клиенты, у которых раньше были отдельные соединения с Internet, после этого могут получать их все по одному единственному кабелю, но и соответственно и быть отключенными от всех них сразу.

Internet-провайдеры также нередко рекомендуют устанавливать "избыточные линии связи" или "резервные соединения", преимущества которых являются весьма сомнительными. При более тщательном изучении может оказаться, что да, кабелей действительно два, но они находятся в одной и той же цепи, или что по резервному каналу уже передается объемный ATM-трафик. Для уверенности в том, что избыточные каналы оправдывают себя, проводите со своими Internet-провайдерами ежегодную проверку.

Проблемы с безопасностью

Об обеспечении безопасности системы подробно рассказывалось в главе 20. Однако здесь тоже стоит коснуться этой темы, потому что вопросы безопасности влияют на очень многие из выполняемых системных администратором задач. Ни один аспект стратегии управления организации не должен разрабатываться без учета безопасности. В главе 20 по большей части рассказывалось о способах предотвращения проблем с безопасностью. Однако продумывание способов восстановления системы после произошедшего из-за связанных с безопасностью проблем инцидента является не менее важным.

Кража данных с Web-сайта относится к числу особенно неприятных инцидентов. Для системного администратора, работающего в компании, которая занимается предоставлением услуг Web-хостинга, подобный инцидент может превратиться в настоящую катастрофу, особенно если речь идет о сайтах, использующих данные кредитных карт пользователей. Наверняка будет поток телефонных звонков от клиентов, от средств массовой информации, от VIP-клиентов компании, которые только что услышали о произошедшей краже в новостях. Кто будет отвечать на эти звонки? Что он должен говорить? Кто будет главным? Кто и что будет делать? Тем системным администраторам, которые работают в очень популярных компаниях, обязательно следует продумать такой сценарий, подготовить подходящие ответы и план действий и возможно даже провести тренировочную проверку, чтобы отработать детали.

На сайтах, принимающих данные кредитных карт, всегда есть заверенные юридически требования к поведению в случае подобных атак. Поэтому системный администратор обязательно должен привлечь к планированию мероприятий на случай проблем с безопасностью сотрудников юридического отдела своей организации и позаботиться о наличии номеров телефонов и имен людей, которым следует звонить во время кризиса.

Когда в новостях объявляют, что такой-то Web-сайт был атакован хакерами и на текущий момент страдает от проблем, возникает ситуация, напоминающая аварию на доро-

ге: все бросаются посмотреть, что же произошло, и в результате Internet-трафик сильно возрастает, нередко настолько сильно, что все, что администраторам только наконец-то с таким трудом удалось исправить, рискует снова оказаться под угрозой. Поэтому если Web-сайт не рассчитан на 25-процентное (или более высокое) увеличение трафика, следует позаботиться о наличии программ балансирования нагрузки, которые будут просто направлять превышающие норму вызовы на сервер, а тот будет возвращать страницу со следующим сообщением: "Извините, сервер слишком загружен и поэтому в данный момент не может обработать ваш запрос".

Истории из "вторых рук", связанные со Всемирным торговым центром

Нам довелось услышать несколько поучительных историй, имевших место 11 сентября 2001 г., в день атаки на Всемирный торговый центр в Нью-Йорке. К сожалению, системные администраторы, которые принимали непосредственное участие в этих историях, не смогли получить у своего руководства разрешения на их разглашение. Но мы решили не сдаваться и включить в нашу книгу хотя бы несколько историй из, так сказать, "вторых" и "третьих рук", которые тоже являются не менее поучительными. Их, конечно, нельзя назвать наилучшими примерами поведения в случае аварий, но зато они все равно очень интересны.

Один Internet-провайдер маршрутизировал все звонки и сетевой трафик через устройство, которое находилось в одном из небольших зданий торгового комплекса. Это здание не пострадало при атаке, поэтому устройство продолжало работать от аварийных генераторов, хотя и нагрелось после того, как была отключена система кондиционирования. К сожалению, администраторы провайдера никак не могли попасть в здание и перезарядить аккумуляторы генераторов. Вывод таков: следует проверять, хватит ли емкости аккумуляторов хотя бы на несколько дней в случае длительных аварийных отключений.

Другая организация, располагавшаяся в одной из башен ВТЦ, держала свои резервные системы во второй башне. К счастью, незадолго до событий руководство решило переместить их в другое место. Третья организация смогла восстановить свою систему, поскольку все важные данные реплицировались на несколько устройств, расположенных в разных местах. Правда, пропало много бумажной документации, так как не все хранилось в электронном виде.

Одна финансовая компания, размещавшаяся в башнях-близнецах, отдавала резервные копии своих данных на хранение в другое место. Однако она не знала, что это "другое место" тоже находилось в одной из башен-близнецов. Так что знать, где хранятся резервные копии данных тоже важно.

Web-сайт компании CNN 11 сентября столкнулся с таким трафиком, который даже невозможно было замерить. Телефонная сеть Нью-Йорка была перегружена, а телевизионные станции, вещавшие через антенны на башнях ВТЦ, оказались парализованными. Все ринулись в Internet, и в результате сайт `www.cnn.com` подвергся немыслимому испытанию. Сначала Web-сайт просто вообще не справлялся с нагрузкой. После попыток распределить нагрузку между несколькими серверами компания CNN, в конце концов, просто уменьшила объем своей начальной страницы, оставив на ней одну картинку и немного текста без каких-либо гиперссылок. Через несколько часов сайт снова стал работать нормально и справляться с нагрузкой.

Испытав такой сильный скачок в объеме Web-трафика, компания CNN внесла ряд изменений в свою стратегию администрирования касательно того, когда следует упрощать начальную страницу, и кто должен принимать такое решение. Таким образом, власть и ответственность перешла от руководящей верхушки к простым системным администраторам.

30.11. Письменные стратегии

Работая над этой главой, мы побеседовали с профессионалами в области системного администрирования, компьютерной безопасности, стандартизации и компьютерного права. И нас поразило их единодушие: все они считают, что документально оформленные и доведенные до сведения сотрудников правила работы совершенно необходимы для нормального функционирования любой организации.

Нужно, чтобы в организации существовали такие документы:

- правила административного обслуживания;
- перечень прав и обязанностей пользователей;
- перечень прав администраторов (пользователей с особыми привилегиями);
- правила создания "гостевых" учетных записей.

Правила работы могут оформляться в виде контрольных списков или инструкций. Они полезны как для администраторов-новичков, так и для "профи". Вот основные преимущества, получаемые при их использовании:

- рутинные задачи всегда выполняются одинаково;
- наличие перечня стандартных процедур снижает вероятность механических ошибок;
- работа по инструкции выполняется администратором гораздо быстрее;
- корректность действий администратора можно сверять с неким эталоном.

Сегодня в корпоративной среде Linux-системы вытеснили мэйнфреймы, и на них лежит основная вычислительная нагрузка. В крупных организациях решение основных задач документировано в виде контрольных списков, часто называемых *книгами инструкций*. Эти книги доступны в интерактивном режиме или хранятся в печатном виде. Составлением и проверкой инструкций занимается дополнительная группа системных администраторов, не входящая в основной штат администраторов, обслуживающих технику и, собственно, использующих эти инструкции. Несмотря на привлечение дополнительных штатных сотрудников, такой подход в итоге окупается.

Перечислим основные задачи, решение которых можно систематизировать в виде перечня инструкций:

- подключение компьютера;
- добавление пользователя;
- конфигурирование локального компьютера;
- настройка процедур резервного копирования для нового компьютера;
- защита нового компьютера;
- перезапуск сложных компонентов ПО;
- перезагрузка Web-серверов, которые не отвечают на запросы или "зависли";

- очистка очереди на печать и перезагрузка принтера;
- модернизация операционной системы;
- установка заплат;
- установка пакета прикладных программ;
- установка программного обеспечения по сети;
- обновление наиболее важных программ (`sendmail`, `gcc`, `named` и т.д.);
- резервное копирование и восстановление файлов;
- списывание резервных лент;
- аварийный останов системы (всех компьютеров, всех, кроме самых важных, и т.д.).

Некоторые проблемы решаются как на уровне правил, так и на уровне инструкций, например:

- кто может иметь учетную запись и как ее создавать?
- что должно делаться при увольнении сотрудника?

Такие вопросы следует регламентировать особенно тщательно, чтобы не стать жертвой известной уловки четырехлетних детей: “Мама не разрешила, надо спросить у папы!”. Часто предложения, начинающиеся со слова “если” представляют собой стратегии, а предложения, начинающиеся со слова “как” — инструкции.

Некоторые стратегические решения зависят от особенностей используемого программного обеспечения или же правил, установленных другими внешними организациями (например, организациями, предоставляющими Internet-услуги). Соблюдение некоторых правил является обязательным, особенно если требуется обеспечить конфиденциальность пользовательских данных. Мы называем такие стратегические решения и правила “безоговорочными”.

В частности, мы считаем, что управление всеми IP-адресами, именами компьютеров, идентификаторами пользователей и групп, а также регистрационными именами должно осуществляться единообразно в рамках всей организации. Некоторые организации (например, транснациональные корпорации) слишком велики для реализации такого подхода, но если его удастся внедрить, это существенно упростит администрирование всей системы. Мы знаем одну компанию, которой удалось реализовать этот подход для 35 000 пользователей и 100 000 компьютеров, так что порог, после которого организация становится слишком большой для него, является в принципе довольно высоким.

Некоторые важные вопросы касаются не только локальной группы администраторов. К их числу относятся:

- действия в случае взлома системы;
- правила экспорта файловых систем;
- критерии выбора паролей;
- удаление учетных записей;
- распространение материалов, защищенных авторскими правами (например, MP3- и DVD-дисков);
- программное пиратство.

Наличие хороших отношений между административными группами в крупных организациях может предотвращать проблемы и создавать атмосферу доверия и сотрудни-

чества. Толчком к налаживанию таких отношений может послужить корпоративная вечеринка. Некоторым администраторам наладить отношения помогают системы обмена мгновенными сообщениями типа ICQ. Такие системы могут приводить к чрезмерному общению сотрудников на работе, но при правильном использовании способны сделать рабочий процесс организации более ровным, особенно если некоторые из сотрудников работают дома или другом офисе. Пересылаемые при помощи таких систем сообщения могут перехватываться, поэтому никакую важную информацию об организации или сети в них лучше не включать.

Правила безопасности

Что конкретно нужно защитить? Данные? Оборудование? Возможность быстро восстанавливать систему после аварии? При разработке правил безопасности необходимо найти разумный баланс между:

- предлагаемыми услугами и обеспечиваемой безопасностью (чем больше услуг, тем меньше безопасность);
- удобством использования и безопасностью (безопасность прямо противоположна удобству);
- стоимостью средств защиты и убытками от ее нарушения.

RFC2196 — это 75-страничный документ, называющийся *Site Security Handbook* (руководство по защите систем) и выпущенный в 1997 г. организацией IETF. В нем содержатся рекомендации системным администраторам по различным аспектам безопасности, правилам и методикам работы пользователей. В этом документе не приводятся пошаговые инструкции по защите Internet-хостов, но зато есть другая, не менее ценная информация. Последние 15 страниц содержат прекрасный перечень интерактивных и печатных ресурсов.

Документ RFC2196 рекомендует включать в свод правил следующие разделы.

- *Рекомендации по закупке оборудования и программного обеспечения.* Участие системных администраторов в выборе оборудования и программного обеспечения может оказать большую пользу, так как часто они знают о недостатках оборудования и программных ограничениях то, чего не афишируют производители.
- *Правила конфиденциальности.* Определяют степень административного контроля над электронной почтой и действиями пользователей, а также правила размещения пользовательских файлов.
- *Правила доступа.* Определяют, кто имеет доступ в систему, что можно делать в рамках прав доступа, какое программное и аппаратное обеспечение можно устанавливать и т.д. Раздел должен включать те же предостережения относительно санкционированного использования и степени контроля, что и правила конфиденциальности.
- *Права и обязанности.* Содержат описание прав и обязанностей пользователей и системных администраторов. Должны четко оговаривать, кому разрешено просматривать сетевой трафик, читать сообщения пользователей и изучать другие подобные по секретности данные. Также должны включать перечень обстоятельств, при которых могут предприниматься такие действия.
- *Правила аутентификации.* Содержат рекомендации по выбору паролей и устанавливают порядок удаленного доступа.

- *Правила доступности*. Определяют, когда система должна быть доступна, когда в ней запланированы обслуживающие процедуры, какие действия следует предпринимать при возникновении проблем и как быстро можно ожидать их устранения.
- *Правила сопровождения*. Определяют график техобслуживания и порядок доступа в систему сторонних специалистов по техобслуживанию.

Примечательно, что из документа RFC2196 исключен раздел, касающийся правил авторизации, которые определяют, кто может создавать новые учетные записи и расширять привилегии. В исходной версии документа *Site Security Handbook* (RFC1244) содержался список конкретных административных задач и вопросов, требующих регламентирования, а не просто описание типов правил. Новая спецификация включает рекомендации для каждого типа службы, которая может работать на компьютере, описывает круг возможных проблем, связанных с этими службами, и методы их решения.

Документ ISO 117999 является более новым источником и уже широко применяется в современной компьютерной индустрии. COBIT — это еще один популярный стандарт (который охватывает не только вопросы безопасности).

Правила должны быть четко сформулированы, документально оформлены и подписаны всеми пользователями и администраторами. Правила применяются ко всем, кто работает в системе, даже если это клиенты, которые платят за предоставляемые им услуги. Если идти на уступки и делать исключения, это ослабит правовую силу правил и подорвет уважение к ним.

Соглашения о правилах работы пользователей

На факультете вычислительной техники университета штата Колорадо используется специальный регистрационный интерпретатор, который, прежде чем впустить пользователя в систему, предлагает ему прочесть и принять соглашение о правилах работы в системе. Только согласившись с требованиями соглашения, пользователь получает доступ к реальному интерпретатору команд. Такая схема доступа очень удобна и экономна, но прежде чем применить ее в своей организации, посоветуйтесь с юристом.

В соглашении о правилах работы необходимо отразить следующие моменты:

- использование учетных записей совместно с друзьями и родственниками (мы рекомендуем никогда не делать этого);
- применение программ дешифровки паролей к локальным файлам паролей;
- применение программ дешифровки паролей к файлам паролей других систем;
- нарушение нормального процесса работы системы;
- проникновение в чужие учетные записи;
- неправильное использование или поделка электронной почты;
- общее (совместное) пользование электронной почтой и электронными средствами массовой информации;
- просмотр файлов других пользователей (можно ли их читать и перезаписывать, одобряется ли это);
- загрузка программ из Internet (запрещена? разрешена? разрешена при условии проверки?);
- обязательное использование брандмауэра и анти-вирусного ПО на компьютерах Windows;

- использование системных ресурсов (принтеров, дисков, модемов, процессора);
- копирование материалов, защищенных авторскими правами (программного обеспечения, музыки, кинофильмов и т.д.);
- выдача разрешений на копирование материалов, защищенных авторскими правами, я другим лицам;
- использование ресурсов для личных или коммерческих целей;
- незаконная деятельность (мошенничество, клевета и т.д.);
- вовлечение в деятельность, которая является разрешенной в одних и запрещенной в других местах (например, распространение порнографии, политическая агитация).

Однако следует четко осознавать, что никакое соглашение не сможет заставить человека не нарушать перечисленные в нем условия. Соглашение можно использовать в качестве основания для привлечения нарушителя к ответственности, для запрещения доступа его учетной записи, для увольнения или исключения этого человека либо для принятия по отношению к нему любых других предусмотренных в данной организации в таких случаях мер.

Ниже в качестве примера приводится короткое и простое соглашение, которое каждый студент факультета компьютерных исследований в университете Мельбурна должен подписать, прежде чем он будет допущен к установленным в этом университете компьютерам.

Я, нижеподписавшийся, настоящим заявляю, что буду соблюдать следующие правила:

- *Обязуюсь использовать вычислительные и сетевые ресурсы факультета исключительно для учебных целей, относящихся к моему обучению компьютерным дисциплинам.*
- *Понимая, что факультет предоставляет учетную запись исключительно для применения ее получателем, обязуюсь не способствовать использованию моей учетной записи и файлов другими лицами и не сообщать свой пароль другим лицам.*
- *Обязуюсь не пытаться получить доступ ни к одному компьютеру, учетной записи, сети или файлу без соответствующего и явного разрешения. Такой доступ является незаконным и противоречит университетским правилам. Если мне станет известно, что такой доступ имел место, я немедленно проинформирую об этом руководство факультета.*
- *Осознаю, что некоторые программы и данные, находящиеся в файловых системах, к которым я могу получать доступ, могут быть защищены законом об авторских правах и другими законами или лицензионными либо другими соглашениями. Обязуюсь не нарушать налагаемых ими ограничений.*
- *Обязуюсь не использовать университетские ресурсы для получения, разработки, запуска и распространения нелицензионного программного обеспечения.*
- *Обязуюсь сохранять конфиденциальность любых полученных мною от университета сведений о программном обеспечении (включая методы и принципы его использования), лицензированном для применения на компьютерах университета, и тем самым обезопасить университет от претензий любого рода, связанных с разглашением мною этой информации в нарушение данного обязательства.*
- *Обязуюсь проявлять предельную честность и порядочность во всех вопросах, связанных с использованием мною вычислительных и сетевых ресурсов факультета; избегать любых действий, связанных с использованием вычислительных и сетевых ресурсов факультета, которые могут повредить репутации факультета или университета.*

Обязуюсь также придерживаться Инструкции 8.1.R7 университета Мельбурна (изложена в Журнале регистрации студентов), которая дополнительно определяет и регулирует порядок использования мною вычислительных и сетевых ресурсов университета.

Сознаю, что действия, противоречащие изложенным выше правилам, повлекут за собой жесткие взыскания, включая отказ в изучении темы или предмета, временный запрет или лишение доступа к университетским вычислительным средствам, временное или полное исключение из университета, штраф и/или другие меры, предусмотренные Законом о компьютерных преступлениях от 1988 г[8].

Обратите особое внимание на неоднозначные слова о честности, порядочности и необходимости беречь репутацию университета. Подобные требования общего характера помогают охватить вопросы, которые трудно описать строгими, детерминированными правилами. И хотя юридическая сила таких требований невелика, все же полезно включать их в свод внутренних правил организации.

Соглашения о правилах работы системных администраторов

В соглашении о правилах работы системных администраторов должны быть перечислены правила пользования привилегиями суперпользователя и соблюдения конфиденциальности пользовательских данных. Руководители IT-отделов должны удостовериться в том, что системные администраторы понимают и поддерживают условия этого документа. В то же время, конечно, бывает трудно ответить на жалобу пользователя о том, что почта не работает, не просмотрев вернувшиеся к нему сообщения (хотя копии заголовков в большинстве случаев вполне хватает для того, чтобы определить суть проблемы и устранить ее).

Если для привилегированного доступа в организации используется такая программа как `sudo`, очень важно, чтобы администраторы выбрали себе надежные пароли и не пользовались своими учетными записями совместно *ни с кем* другим. Неплохо будет организовать регулярную проверку паролей системных администраторов с помощью программы John the Ripper. Также очень важно, чтобы администраторы никогда не выполняли команду `sudo sh` и не заходили в оболочку после того, как откроют свой любимый текстовый редактор при помощи `sudo`: такие способы использования команды `sudo` являются некорректными и сводят на нет смысл функции ведения журналов `sudo`. Регистрационный журнал существует не для того, чтобы следить за администраторами, а для того, чтобы помогать восстанавливать цепочку событий в случае, если что-нибудь пойдет не так.

🕮 Об утилите `sudo` более подробно рассказывалось в разделе 3.4, а об утилите John the Ripper — в разделе 20.10.

У некоторых системных администраторов желание демонстрировать свои привилегированные возможности преобладает над здравым смыслом. Таким администраторам лучше мягко, но уверенно предложить сменить работу.

[8] Учтите, что здесь речь идет об Австралии. Аналогичные законы, касающиеся компьютеров и программного обеспечения, существуют и в США.

В некоторых организациях обладание паролем суперпользователя является показателем занимаемого положения и иногда даже ценится больше, чем наличие ключа от уборной начальства или доступа к близко расположенному месту на автостоянке. Часто бывает так, что этим паролем обладают инженеры, которым он совсем не нужен или у которых его быть не должно. На одной известной нам фирме поступили следующим образом: всем инженерам предложили пароль суперпользователя при условии, что его обладатели будут носить пейджер и в случае необходимости помогать всем, кто их попросит. Количество желающих обладать этим паролем резко снизилось.

Есть еще один успешно апробированный нами метод: поместить пароль суперпользователя в конверт и спрятать его в месте, доступ к которому имеют только системные администраторы. Для выполнения работы администраторам обычно достаточно утилиты `sudo`; но если им действительно по какой-то причине понадобится именно пароль суперпользователя, они смогут вскрыть конверт. После этого они должны будут изменить пароль суперпользователя и сделать новый конверт. Процедура довольно проста и в то же время достаточно утомительна для того, чтобы стимулировать администраторов пользоваться все-таки утилитой `sudo`. Важно, чтобы старые пароли тоже хранились какое-то время где-нибудь поблизости на случай, если вдруг окажется, что на момент ввода нового пароля какие-то из компьютеров не работали или были отключены от сети.

Если пароль суперпользователя известен всем сотрудникам, он должен меняться каждый раз, когда кто-то из сотрудников увольняется. В организации может существовать множество административных паролей: для компьютеров, принтеров, сетевого оборудования, PROM или BIOS, баз данных и т.д. Поэтому следует сесть и написать перечень всех паролей, которые должны изменяться (и как они должны изменяться) в случае увольнения сотрудников.

30.12. Правовые вопросы

Федеральным правительством США и некоторыми штатами были изданы законы о компьютерных преступлениях. На федеральном уровне до недавнего времени существовало два закона, а теперь их четыре:

- Федеральный Закон о конфиденциальности связи (The Federal Communications Privacy Act);
- Закон о компьютерном мошенничестве и компьютерных злоупотреблениях (The Computer Fraud and Abuse Act);
- Закон о компьютерных кражах (The No Electronic Theft Act);
- Закон об авторских правах (Digital Millenium Copyright Act).

С началом нового тысячелетия основными проблемами стали ответственность системных администраторов, сетевых операторов и провайдеров, предоставляющих услуги Web-хостинга; надежное шифрование в системах электронной коммерции, одноранговые сети и угроза, которую они представляют для индустрии развлечений; защита авторских прав и защита конфиденциальности. Рассматриваемые в этом разделе темы затрагивают как эти вопросы, так и ряд других правовых аспектов, имеющих отношение к системному администрированию.

Шифрование

Необходимость шифрования в сферах электронной коммерции и компьютерных коммуникаций очевидна. Однако в некоторых странах шифрование запрещено законом. Полиция и другие организации, охраняющие закон, не хотят, чтобы граждане могли обмениваться данными, которые они (в полиции) не смогут расшифровать.

В США законы, касающиеся шифрования, сейчас изменяются. Раньше, например, запрещалось экспортировать промышленные технологии шифрования. Поэтому компаниям приходилось разрабатывать по две версии программ шифрования: одну для локального рынка, а другую, усеченную — на экспорт. Одним из побочных эффектов такой политики было то, что многие проекты связанного с шифрованием ПО начали создаваться в других странах. Абсурдность такой патентной политики (в остальных странах мира технологиями шифрования пользуются уже давно) и потребности электронной коммерции постепенно вынудили правительство изменить свою точку зрения.

Хотя некоторые ограничения на экспорт по-прежнему существуют, введенные в 2004 г. изменения значительно улучшили ситуацию в США. Какие именно это были изменения можно узнать по адресу `www.bis.doc.gov/Encryption`.

Организация IETF работает над стандартами в области защиты двусторонних коммуникаций на уровне протоколов (проект IPsec), а производители начали разрабатывать системы на основе этих стандартов. Аутентификационная часть пока еще привязана к конкретной системе, однако шифрующее программное обеспечение уже часто устанавливается отдельно. Такая архитектура позволяет сохранить гибкость, необходимую для стран, где шифрование запрещено.

На одном из состоявшихся не так давно (в 2006 г.) заседаний Верховного суда США, было принято решение о том, что код программного обеспечения должен считаться правом человека на свободу слова и, следовательно, защищаться первой поправкой Конституции США. В 1990 г., будучи студентом университета Беркли, Дэн Бернштайн (Dan Bernstein) разработал новый алгоритм шифрования. Однако Государственный департамент США запретил ему сообщать о своем открытии общественности, классифицировав его изобретение как “военное имущество”. Защищать права Дэна взялась организация Electronic Frontier Foundation (`eff.org`), которая в конце концов все-таки выиграла это дело, потратив на него “всего лишь” каких-то 16 лет!

Авторские права

Музыкальные и кинопроизводители начали с ужасом замечать, что проигрывать музыку с компакт-дисков и просматривать фильмы в формате DVD стало возможно на домашних компьютерах. Для них это, с одной стороны, открывает новые возможности, а с другой — представляет угрозу. Что из этого получится в итоге зависит от того, как они к этому отнесутся: прогрессивно или агрессивно; к сожалению, они похоже пошли по второму пути.

Содержимое DVD-дисков шифруется по технологии CSS (Content Scrambling System — система шифрования содержимого). Это делается для того, чтобы диски могли воспроизводиться только лицензированными и одобренными проигрывателями. Эти проигрыватели, как и лицензионное программное обеспечение для воспроизведения, поставляемое вместе с DVD-дисководами, имеют ключ для декодирования дисков.

Один норвежский студент осуществил инженерный анализ процесса шифрования CSS и опубликовал в Internet программу, названную DeCSS. Эта программа не обходит

схему шифрования DVD, а просто использует ключ легального проигрывателя, имеющегося в Windows, для декодирования потока данных и сохранения расшифрованных данных на диске с целью их последующего воспроизведения в Linux.

Организации MPAA (Motion Picture Association of America — Ассоциация кинокомпаний США) и DVD Copy Control Association тут же подали в суд на множество "дистрибьюторов" программы DeCSS. Таковым посчитали всякого, чей Web-сайт содержал ссылку на копию программы. Подсудимые обвинялись не в краже материалов, защищенных авторскими правами, а в распространении коммерческой тайны и "нарушении защиты от копирования", что запрещено законом США "Digital Millennium Copyright Act" (DMCA) от 1998 г[9]. Организация MPAA выиграла процесс, и программа DeCSS была признана нарушающей закон DMCA. На текущий момент (2006 г.) использование любого DVD-проигрывателя с открытым исходным кодом считается в США незаконным.

Недавно еще две компании пытались использовать связанные с авторскими правами и запретом на обратную инженерию пункты закона DMCA, чтобы "придушить" своих конкурентов: компания Sears пыталась добиться запрета на использование в запатентованных ею подъемных гаражных дверях устройств для открывания сторонних производителей, а компания Lexmark — запретить сторонним компаниям повторно заправлять картриджи Lexmark для лазерных принтеров и перепродавать их по более дешевой по сравнению с новыми цене. В обоих случаях интересы компании представлял один и тот же адвокат из организации EFF и в обоих случаях он выиграл дело.

Еще одно связанное с законом DMCA дело — MGM против Grokster и Streamcast — дошло до Верховного Суда США. Предыдущие решения (например, по делу Sony против Betamax в 70-х гг. прошлого столетия) гласили, что продажа технологий, способных нарушить защиту чьего-нибудь запатентованного материала, не является незаконной; виновными могли быть признаны только те, кто пользовался этими технологиями непосредственно для этих целей. В деле Grokster компания Streamcast зарабатывала деньги тем, что рекламировала свое программное обеспечение как позволяющее делать копии защищенной авторским правом музыки и кинофильмов. Это поощрение к нарушению авторских прав было признано нарушением закона DMCA: 9 из 9 судей проголосовали в пользу MGM!

Одноранговым сетям типа BitTorrent подобное обвинение в "поощрении" вроде бы пока не грозит, потому что многие из используемых в них для обмена файлами приложений не подразумевают пересылку материала, защищенного авторскими правами. Однако детали все равно пока остаются довольно неясными.

Компания SCO постоянно "трусит" поставщиков Linux за использование в своих дистрибутивах ее защищенных авторскими правами фрагментов кода UNIX. Она возбудила уже тысячи дел против отдельных индивидуумов и одно даже против IBM. Организация ODSL (Open Source Development Labs — Центр по разработке ПО с открытым исходным кодом) учредила законный фонд в 10 миллионов долларов для защиты пользователей Linux. Ее поддержали такие компании, как Cisco, Dell, Ericsson, Fujitsu, Hitachi, HP, IBM, Intel, Nokia, Red Hat, Sun и Toshiba. Недавно (в июле 2006 г.) суд поддержал сторону IBM, признав свыше 250 претензий SCO слишком общими и, раз так, не подлежащими дальнейшему рассмотрению; подробности можно узнать на сайте `groklaw.net`. Такое решение суда свидетельствует о том, что количество возбужденных SCO дел уже

[9] Закон DMCA запрещает "нарушать" или "обходить" меры, предназначенные для защиты цифрового содержимого, даже если причина нарушения подпадает под льготы на "свободное использование", предусмотренные в традиционном законе об авторских правах. Не нужно даже и говорить, что это изменение является серьезным ограничением льгот на свободное использование.

превысило все мыслимые пределы, и должно отбить последующую охоту возбуждения дел по поводу и без повода.

Конфиденциальность

Обеспечение конфиденциальности всегда было непростой задачей, а с ростом популярности Internet ситуация и вовсе усложнилась. Возьмем, к примеру, недавний инцидент в университете штата Мичиган. Медицинские карточки студентов университета в один прекрасный день оказались опубликованными в Internet. К тому времени, как один из студентов обнаружил утечку информации, карточки находились в сети уже несколько месяцев. И это не единичный случай. Сегодня очень часто взламываются базы данных с номерами кредитных карт. Почти ежедневно в электронный почтовый ящик приходят фальшивые письма якобы из банка с просьбой проверить и подтвердить данные счета в связи с какими-то проблемами. Но при более подробном анализе такого письма в текстовом, а не в Web-формате, оказывается, что все данные пойдут не в банк, а к какому-нибудь хакеру в Восточной Европе или Азии.

Нарушение конфиденциальности компанией DoubleClick

Еще один связанный с нарушением конфиденциальности скандал разразился вокруг компании DoubleClick — рекламного агентства, создающего баннеры для Web-страниц. Годами эта компания заверяла своих пользователей в их полной анонимности, утверждая, что никто их не отслеживает и не пытается идентифицировать. Однако недавно она приобрела фирму, которая таки занимается разработкой средств поиска и анализа данных, и начала собирать информацию о каждом пользователе, который посещал Web-страницу, содержащую баннер DoubleClick. Конечно, как только об этом стало известно, поднялась шумиха, и компании пришлось закрыть проект и нанять пару опытных в таких делах юристов.

Нарушение конфиденциальности компанией Sony

Компания Sony включала во многие свои CD-диски программу под названием *XCP* (eXtended Copy Protection) для дополнительной защиты от копирования. Эта программа автоматически устанавливалась на компьютере владельца такого CD-диска, скрывала свои собственные файлы, так чтобы пользователь их не заметил, и затем отправляла компании Sony отчеты о прослушиваемой им музыке. Т.е. программа XCP по сути являлась разновидностью средства, известного в профессиональных кругах как “руткит”, а раз так — устанавливаемые ею файлы могли делать машину пользователя открытой для хакеров; подробнее об этом рассказывалось в разделе 20.9.

Однажды некий Марк Руссинович (Mark Russinovich) приобрел CD-диск производства Sony с защитой DRM (Digital Rights Management — управление правами на цифровые материалы) на `amazon.com` и установил его на своей системе Windows. Позже он обнаружил программу XCP и проанализировал ее природу и масштабы ее воздействия.

“Программное обеспечение, которое Sony без моего ведома установила на моей системе, не только применяет технологии, которые обычно используются во вредоносном ПО, для сокрытия своего присутствия, но еще и плохо написано и не предоставляет никаких средств для деинсталляции. Хуже того, в случае обнаружения его скрытых файлов антивирусной программой пользователи рискуют повредить свой компьютер, если попытаются просто взять и удалить эти скрытые файлы.

Я вполне согласен с тем, что производители имеют право использовать механизмы для защиты своей продукции от незаконного копирования, но, на мой взгляд, правильный баланс между свободным использованием и защитой от копирования все-таки пока еще не найден. Поведение компании Sony является ярким примером злоупотребления технологией DRM."

О наличии программы XCP и том, что ее нельзя деинсталлировать, ничего не сказано в лицензионном соглашении (EULA), которое пользователи должны подписать, чтобы установить CD на своем компьютере[10]. В Великобритании это является явным нарушением закона о неправильном использовании компьютерных технологий "Computer Misuse Act". В штате Калифорния это тоже является нарушением закона "California Business & Protections Code" и карается штрафом в размере 1000 долларов за каждый пострадавший из-за этого компьютер. Организация EFF подала коллективный иск против Sony с требованием возместить ущерб; подробности — на сайте `www.sonybmgcdtechsettlement.com`.

А почему это должно волновать системных администраторов? Ну, если программа XCP от Sony установлена на настольных компьютерах по всей организации, все данные этой организации рискуют оказаться в "чужих руках".

Запись звонков и Web-данных

Три самых крупных телефонных компании в США были обвинены в предоставлении телефонных и Internet-записей своих клиентов (каковых было около 200 миллионов) Агентству национальной безопасности (National Security Agency, NSA). Qwest — еще одна крупная американская телефонная компания — отказалась сотрудничать с NSA из соображений конфиденциальности. Она отказалась это делать даже после того, как NSA не двузначно намекнула, что из-за этого Quest больше не будет получать никаких засекреченных правительственных контрактов. И правильно сделала, потому что организация EFF сейчас подала на три остальные телефонные компании в суд.

История показывает, как записи телефонных звонков могут использоваться в незаконных целях. В 2002 г. компьютер колумбийского наркокартеля был захвачен полицией[11]. Анализ содержащихся на жестких дисках этого компьютера данных показал, что в 1994 г. записи звонков от телефонных посредников использовались для определения "расстояния" сотрудников картеля от представителей правоохранительных органов. Те из сотрудников, которые находились "слишком близко", были устранены. Анализ трафика является очень мощным средством и многие им пользуются.

Несколько крупных поисковых служб и Web-сайтов Internet-портала (например, Yahoo!) предоставляли правительству США данные об искомой пользователями в Web информации. Компания Google единственная не подчинилась правительственным требованиям. Однако она все-таки создала специальную поисковую службу для правительства Китая, которое захотело иметь возможность подвергать цензуре результаты Web-поисков в Китае. Сейчас сообщают, что Google пересматривает это свое решение.

Сотовая телефонная сеть в США тоже вызывает недовольство по некоторым связанным с конфиденциальностью причинам. Когда человек путешествует по стране с сотовым телефоном в кармане, его телефон автоматически регистрируется в локальной

[10] "EULA" расшифровывается как "End-User License Agreement", что дословно переводится так: лицензионное соглашение для конечного пользователя.

[11] Подробнее об этом можно узнать по адресу `amsterdam.nettime.org/Lists-Archives/nettime-l-0207/msg00015.html` .

сотовой сети каждого места, которое он проезжает, для того, чтобы ему могли перенаправляться его звонки. К сожалению, правительству не нужен ни ордер, ни даже достаточные основания для того, чтобы отследить местонахождение этого человека по его сотовому телефону.

Лежащим в основе правовым вопросом является определение так называемых "автоматически регистрируемые данные" (pen-register data). В контексте телефонов под этими данными подразумевается информация о том, кто звонил, кому звонил и когда, не включая фактического содержания разговора. В 1984 г. вышел Закон о конфиденциальности электронной связи (Electronic Privacy Communications Act), в котором указывалось, что автоматически регистрируемые телефонные данные должны отделяться от данных фактического содержания телефонных разговоров (и те, и другие из которых ранее защищались одинаково), и что телефонным компаниям разрешается разглашать данные первого типа в случае соответствующего распоряжения суда, получить которое не представляло труда. В 2001 г. вышел Закон о патриотизме (Patriot Act), в котором понятие "телефонные номера" было расширено и стало включать себя также и такие связанные с ПО и Internet данные, как IP- и URL-адреса.

Т.е. посещаемые пользователями URL-адреса и выполняемые ими Web-поиски теперь классифицируются как автоматически регистрируемые данные и официально не защищаются как конфиденциальная информация. Для того чтобы системный администратор предоставил содержимое электронных сообщений или файлов пользователей, необходим ордер, но для того чтобы получить доступ к содержимому большинства журнальных файлов, достаточно всего лишь соответствующего распоряжения суда.

Еще одним свойством автоматически регистрируемых данных является то, что они могут использоваться и служить доказательством в судебных разбирательствах, даже если были получены представителями правоохранительных органов незаконно. Вот такие сейчас настали страшные времена.

"Прощелкивание" EULA-соглашений

EULA — это сокращенное название лицензионных соглашений для конечного пользователя (End User License Agreements, EULA), условия которых предлагается принять во время установки приобретенного программного обеспечения. Большинство людей не читает их внимательно, а даже если и читает, то все равно щелкает на кнопке Yes (Да, согласен), потому что за программное обеспечение уже заплачено, CD-диск уже открыт и процесс установки уже запущен — слишком поздно возвращать программное обеспечение и требовать обратно свои деньги. А ведь некоторые EULA-соглашения содержат очень злонамеренные требования; ниже перечислены несколько вещей, на которые пользователю может предлагаться согласиться.

- Разрешить производителю осуществлять мониторинг или учет использования ресурсов системы (Red Hat!).
- Не критиковать продукт и производителя (Microsoft).
- Не тестировать продукт и не сообщать результаты (McAfee).
- Отказаться от права внедрения выбранной компанией стратегии конфиденциальности (McAfee).
- Воздерживаться от настройки или ремонта компьютера.
- Разрешить выполнять автоматическое обновление продукта.

- Не пытаться воспроизводить код продукта.
- Не удалять продукт с компьютера.
- Не использовать продукт вместе с конкурирующим продуктом.

И это далеко не весь список, наверняка мы упустили еще много вещей подобного рода...

Внедрение стратегии в жизнь

Журнальные файлы могут не оставлять никаких сомнений в том, кто именно совершил преступление, однако для суда это не доказательство. Для защиты своих интересов в суде следует использовать официально заверенные бумажные документы. Время, указанное в журнальном файле, тоже не всегда является уликой, если только компьютерные часы не синхронизируются по протоколу NTP.

Чтобы иметь возможность обвинить злоумышленника, придется утвердить правила безопасности. Эти правила обязательно должны включать положение вроде:

- *Несанкционированное использование компьютерных систем университета означает нарушение не только университетских правил, но и федеральных законов. Несанкционированное использование является преступлением, влечет за собой уголовную и гражданскую ответственность и подлежит наказанию, предусмотренному законодательством.*

Мы также рекомендуем поместить в файл `/etc/motd` (файл сообщения дня) сообщение, предупреждающее пользователей о действующей политике отслеживания. Наше сообщение выглядит так:

```
Вводимая вами с клавиатуры информация может отслеживаться в случае угрозы
или фактического возникновения инцидента, связанного с безопасностью.
```

Для уверенности в том, что пользователи прочитают это сообщение хотя бы раз, можно включить его в выдаваемые новым пользователям файлы инициализации. Если для регистрации в системе требуется использовать команду `ssh` (а так и должно быть), тогда можно настроить файл `sshd.config` так, чтобы он всегда показывал файл `motd`.

Нужно обязательно указать, что сам факт пользования учетными записями означает согласие с установленными правилами, а также объяснить пользователям, где они могут взять дополнительные экземпляры свода правил, и поместить основные документы на соответствующей доске объявлений. Кроме того, следует перечислить наказания, предусмотренные в случае несоблюдения установленных правил (удаления учетной записи и т.д.).

Контроль = ответственность

Провайдеры Internet обычно следуют правилам пользования сетью, которые диктуют стоящие над ними поставщики и которые являются обязательными для их клиентов. Таким образом, вся ответственность за действия клиентов ложится на самих клиентов, а не на провайдеров. Целью такой стратегии является защита провайдеров от ответственности за спам и прочие незаконные действия, например хранение пользователями на своих хостах незаконного или защищенного авторскими правами материала. В каждом штате и стране имеются свои законы на этот счет.

Предположим, с вашего хоста в Web был отправлен или опубликован какой-нибудь "нехороший" материал. Если вы — пользователь службы CompuServe (которая теперь входит в состав компании AOL), проблем не избежать. В судебном деле "Кабби против CompuServe" рассматривался случай публикации некоего клеветнического сообщения. Судья постановил, что администрация CompuServe невиновна, а модератора группы новостей, в которой это сообщение было опубликовано, обвинил в халатности. Мораль: чем больше вы пытаетесь контролировать информацию, тем большую ответственность на себя берете.

Этот принцип прекрасно иллюстрирует история, произошедшая в Техасе с изобретательным студентом по фамилии Чизер (Cheeser), который организовал свой бизнес. Что он сделал: он написал несколько сценариев на языке Perl, с помощью которых извлекал из групп новостей Usernet непристойные картинки и публиковал их на собственном Web-сайте, взимая с подписчиков по 12 долларов в месяц и, тем самым, зарабатывая весьма неплохие деньги.

Чизер старался не нарушать закон и не подписывался на группы новостей, где могла быть детская порнография. Однако некоторые из используемых им групп новостей все-таки были "на грани", т.е. иногда содержали незаконный материал. Этот просчет, а также тот факт, что он выбрал для своего бизнеса один из консервативных округов штата Техас, привели к тому, что его деятельность вскоре была прекращена.

Получив анонимное сообщение (вероятно, от конкурента), полиция конфисковала его компьютеры. Естественно, они нашли образец детской порнографии, случайно попавший из "безопасной" группы новостей. И хотя дело не дошло до суда, из соглашения о признании вины подсудимым стало ясно, что судья считает Чизера виновным не в создании Web-сайта, а в том, что он плохо отбирал материалы. Так что если бы Чизер не отбирал их вовсе, по закону все было бы в порядке. Вывод: никогда не берите на себя ответственность за отбор порнографии!

Этот принцип распространяется и на другие отношения с внешним миром. С юридической точки зрения вывод однозначен: чем больше вы контролируете использование Internet-ресурсов, тем большую ответственность несете за действия пользователей. Если вы знаете о противоправной деятельности, закон обязывает вас расследовать ее и доложить о результатах куда следует.

По этой причине некоторые компании ограничивают данные, которые должны регистрироваться в журнальных файлах, период времени, на протяжении которого должны храниться журнальные файлы, и объем журнальной информации, которая должна записываться на резервные ленты. Для реализации подобной стратегии существует даже специальное программное обеспечение (например, программа Squid), которое поддерживает различные уровни журнализации, позволяющие системным администраторам устранять проблемы и при этом не нарушать конфиденциальности действий пользователей. Местные законы могут разрешать выбирать только какие-нибудь определенные уровни журнализации, поэтому не забывайте проконсультироваться по этому поводу у юриста.

Системные администраторы должны быть знакомы со всеми установленными в организации или университете правилами и следить за их соблюдением. Нет ничего хуже не соблюдаемых или противоречащих друг другу правил, как с практической, так и с юридической точки зрения.

Лицензии на программное обеспечение

Многие компании оплачивают меньшее количество копий программ, чем используют на самом деле. Если об этом становится известно, компания теряет гораздо больше, чем сэкономила на приобретении недостающего числа лицензий. Другие компании получают демо-версию дорогого пакета и взламывают ее (меняют дату на компьютере, определяют лицензионный ключ и т.д.), чтобы пакет продолжал работать по истечении демонстрационного срока. Как системный администратор должен реагировать на предложения нарушить лицензионное соглашение и установить нелицензионные копии программы на дополнительные компьютеры? Что он должен делать, если обнаруживается, что на обслуживаемых им компьютерах работает пиратское программное обеспечение? И как быть с условно-бесплатными программами, за которые так никогда и не заплатили?

Это сложный вопрос. К сожалению, начальство не всегда поддерживает администратора, предлагающего удалить нелицензионные копии программ либо оплатить их. А ведь часто именно системный администратор подписывает лицензионное соглашение, требующее удалить демонстрационные копии после определенной даты, тогда как решение их не удалять принимает руководитель.

Даже если вы дорожите своей работой, помните, что речь идет о вашей личной и профессиональной честности. Что касается работы, то спрос на хороших системных администраторов настолько велик, что надолго без работы вы не останетесь. Нам известна пара случаев, когда непосредственный начальник системного администратора предлагал ему "не высовываться". И в том, и в другом случае администратор написал докладную записку вышестоящему руководству, в которой указал количество лицензированных и используемых копий, а также процитировал лицензионное соглашение. В одном случае подход сработал и уволен был непосредственный начальник системного администратора. Во втором случае уволиться пришлось системному администратору, потому что даже вышестоящее руководство отказалось действовать по закону. Чтобы вы ни делали в такой ситуации, обязательно делайте это в письменной форме. Требуйте, чтобы ответы предоставлялись в письменном виде, а если не получится, документируйте полученные инструкции в виде коротких служебных записок и отправляйте их ответственному лицу.

Регуляторная политика

Серьезные скандалы, разразившиеся в таких крупных компаниях как Enron и WorldCom (MCI) вокруг финансовых данных, привели к выходу в 2002 г. закона под названием "Sarbanes-Oxley Act" ("Закон Сарбейнса-Оксли"), известного в народе просто как "SOX". В этом законе были перечислены правила ведения бухгалтерского учета для компаний, акции которых котируются на фондовых биржах США, и указывалось, что отныне работающие в этих компаниях бухгалтеры будут лично отвечать за достоверность финансовых данных.

Поскольку большая часть данных компаний хранится в компьютерах, закон SOX оказывает непосредственное влияние на системное администрирование. Некоторые из перечисленных в нем правил (таких как ведение журналов электронной почты, регистрация данных аудита, использование внутренних средств контроля), являются простыми, некоторые же — не очень. Например, закон SOX обязывает определять шаблон нормального функционирования и использования хранилища данных (например, базы данных) и создавать средства для выявления его аномального поведения и получения к нему несанкционированного доступа. Также он обязывает все документировать и доку-

ментально подтверждать то, что внедренные средства контроля используются регулярно. Соблюдение этого закона очень сильно снижает продуктивность и "боевой" дух системных администраторов.

С другой стороны, закон SOX является очень выгодным для тех, кто занимается продажей устройств хранения данных или предоставлением услуг бухгалтерского учета. Большим компаниям соблюдение этого закона обошлось за прошлый год в среднем в 4 миллиона долларов, и в будущем этот показатель будет только расти. Теперь, когда исполнители лично отвечают за нарушения и могут из-за них отправиться в тюрьму, они иногда требуют выполнения новых, довольно странных процедур по системному администрированию, дабы соблюсти все требования закона SOX.

SOX повлиял на работу частных и благотворительных организаций тоже. Писать правила по срокам хранения данных и срокам их удаления стало необходимо как для печатных, так и для электронных документов.

Один системный администратор как-то тестировал в одной крупной компьютерной компании систему защиты на проникновение под руководством международной бухгалтерской фирмы. Он обнаружил, что практически любой мог незаметно проникнуть в базу данных Oracle Financials этой компании и изменить хранящиеся там финансовые данные, т.е. сделать именно то, от чего соблюдение закона SOX по идее должно уберегать. Бухгалтерская фирма захотела, чтобы он подтвердил свои предположения, изменив финансовые данные клиента. Он отказался, но она продолжала настаивать, и тогда он решил доложить о поведении этой фирмы в соответствующий отдел этой компании. В конечном счете, компания разорвала контракт с этой бухгалтерской фирмой, а системный администратор вскоре после этого тоже потерял свою работу.

Вопросы ведения бухгалтерского учета и управления в сфере IT-технологий являются сегодня очень важными. Нормы и псевдостандарты по указанию, оценке и документированию привели к появлению целого ряда аббревиатур, среди которых SOX, ITIL (Information Technology Infrastructure Library — библиотека передового IT-опыта), COBIT и ISO 17799, и это далеко не полный список. К сожалению, от всех этих аббревиатур в головах системных администраторов возникает путаница, а программного обеспечения для реализации всех якобы необходимых согласно последним законам средств контроля на текущий момент просто катастрофически не хватает.

30.13. Патенты на программное обеспечение

Сначала патентное ведомство запрещало патентовать математические теоремы. Затем на смену теоремам пришли алгоритмы, и их тоже нельзя было патентовать. Потом алгоритмы были реализованы в аппаратных средствах, и в таком виде их уж точно нужно было разрешить патентовать. Так патентное ведомство и поступило: оно разрешило патентовать аппаратные средства, но на программное обеспечение это разрешение по-прежнему не распространялось. Однако отказ в выдаче патента можно было оспорить в суде, и случилось так, что один из судов низшей инстанции одобрил патенты на программное обеспечение. Благодаря этому, патентное ведомство, в конце концов, хоть и против своей воли, но начало выдавать такие патенты, в некоторых случаях даже на приложения, о которых было заявлено 10–15 лет назад.

К сожалению, патентное ведомство всегда плохо разбиралось в качестве программного обеспечения и потому выдало множество неуместных (кто-то даже может сказать "глупых") патентов. Например, оно выдало целых пять разных патентов на один и тот

же алгоритм сжатия данных Лемпеля-Зива (Lempel-Ziv). Этот алгоритм первоначально был опубликован в одном математическом журнале и реализован в Berkley UNIX. Концепция включаемого файла запатентована. Концепция курсора запатентована. Вычитание было запатентовано как средство устранения неполадок в программном обеспечении, не совместимом с Y2K. Процесс копирования изображения из памяти в отображаемое на экране окно запатентован, равно как и применение операции XOR (исключающее "ИЛИ") для обработки перекрывающихся окон. Запатентованы и несколько стандартов шифрования данных. Концепция встраивания в пользовательский интерфейс рекламного материала тоже запатентована.

В США закон о патентах гласит, что если конкретное изобретение может быть запатентовано, значит, может быть запатентовано и любое малейшее улучшение этого изобретения. В случае программного обеспечения таким улучшением может считаться любой шаг вперед. Наивность патентного ведомства в вопросах, касающихся программного обеспечения, и неумение отличить уникальное от очевидного, приводят к тому, что каждый год оно выдает тысячи патентов на ПО сомнительной ценности.

Но недавно в США возникла даже еще большая угроза: патенты на используемые в ПО бизнес-методы. Компании убедили патентное ведомство в том, что практически любой метод компьютеризации стандартной бизнес-процедуры является изобретением, которое может быть запатентовано. И патентное ведомство начало выдавать патенты на самые обычные операции, такие как извлечение учетной записи клиента из компьютерной базы данных, когда клиент звонит в службу технической поддержки. Компания Amazon.com получила патент на "технологию одного щелчка", а потом, на его основании, — судебный запрет, требующий, чтобы компании Barnes и Noble позволяли своим клиентам приобретать книги путем выполнения как минимум двух щелчков кнопкой мыши[12].

Европейский Союз, напротив, в июле 2005 г. постановил, что любые "схемы, правила и методы для выполнения интеллектуальных, игровых или бизнес-операций, а также программы для компьютеров" не являются изобретением и потому не могут быть запатентованы. За этот закон проголосовало 648 человек, а против — только 14.

Патентное ведомство США сейчас пытается устранить последствия своего закона, но, похоже, что во многих случаях ущерб уже был нанесен. Одним из ярчайших примеров является произошедший в 1994 г. отзыв принадлежавшего компании Compton New Media патента, который охватывал информационно-поисковые системы для данных, хранящихся на CD-ROM. По мнению некоторых аналитиков, это отразилось на 80% всех существующих продуктов CD-ROM (хотя, пожалуй, эта оценка все-таки немного завышена). В конечном счете, благодаря проведенной производителями программного обеспечения дорогостоящей и занявшей немало времени кампании для демонстрации существования прототипа, каждый из 41 пунктов патента был признан недействительным.

Обнаружение прототипа является настоящим слабым местом в процессе патентного ведомства. Патентные заявки держатся в секрете, а из-за неопытности сотрудников патентного ведомства в вопросах программного обеспечения им очень трудно отличить, какие из заявок действительно представляют новую технологию, а какие — уже существующую. В результате потом начинаются судебные разбирательства, выигрывают от которых только адвокаты.

[12] Компания DoubleClick.net, пожалуй, могла опередить их и здесь, запатентовав технологию двойного щелчка.

Однако в США даже отказ патентного ведомства в выдаче патента на предложенное изобретение не останавливает патентных махинаторов от обращения в суд. Например, компания NTP подавала заявку на получение пяти патентов в такой широкой области, как технологии беспроводной электронной почты. Патентное ведомство отклонило все пять заявок этой компании, причем две последних были отклонены окончательно без права апелляции. Тогда компания NTP, основываясь на этих признанных недействительными и потому отклоненных патентах, взяла и подала в суд на канадскую компанию Research in Motion (RIM), производителя принимающих электронную почту сотовых телефонов BlackBerry.

Очевидно, что преимущество было на стороне компании RIM. Она провела свои собственные исследования и разработала свою собственную технологию, а патентные заявки компании NTP были отклонены патентным ведомством. Тем не менее, судья, слушавший это дело, похоже больше склонялся к тому, чтобы принять сторону NTP и вынести судебный запрет, который бы сделал продажу новых телефонов BlackBerry производства компании RIM на территории США незаконной. Более того, после этого запрета существующие клиенты больше бы не смогли использовать свои телефоны BlackBerry для работы с электронной почтой.

Подсчитав, что последствия вынесения такого запрета будут слишком большими, компания RIM решила уладить дело путем одноразовой выплаты суммы размером 612 500 000 долларов в обмен на постоянную лицензию на использование технологии, представленной в пяти отклоненных патентных заявках компании NTP. То был счастливый день для пользователей BlackBerry вроде нас, но печальный — для небольших компаний с хорошими идеями и хорошими продуктами.

Все подробности можно узнать на странице архивов организации Electronic Frontier Foundation по адресу `www.eff.org`. Неплохим источником сенсационных новостей является посвященный патентам раздел сайта `slashdot.org`.

30.14. Стандарты

Процесс стандартизации в одних случаях помогает (интерфейсы Wi-Fi разных производителей могут работать друг с другом), а в других — вредит (сетевые протоколы OSI стоимостью в миллионы долларов оказались пустой тратой денег). Комитеты по стандартизации должны обобщать и формализовать существующие подходы, не заниматься изобретательством.

Задача стандартов состоит в том, чтобы предоставить пользователям возможность покупать совместимые продукты, созданные конкурирующими производителями. Некоторые организации, вовлеченные в процесс стандартизации, просто систематизируют применяемые методики. Другие — преследуют стратегические цели: они стремятся к победе над конкурентами или сокращению объемов работ по приведению собственных продуктов в соответствие с нормами.

Часто крупнейшими заказчиками стандартизированных систем и приложений являются государственные учреждения. Наличие стандартов позволяет им не привязываться к конкретным производителям. Однако крупные компании могут по собственной инициативе замедлять процесс стандартизации в ожидании, пока их продукты не закрепятся на рынке.

Существует несколько комитетов по стандартизации, как официальных, так и неофициальных. В каждом из них действуют свои правила приема новых членов, голосования

и работы. С точки зрения системного или сетевого администратора самыми важными комитетами являются LSB (Linux Standard Base) организации Free Standards Group, POSIX (Portable Operating System Interfaces) и IETF (Internet Engineering Task Force; см. раздел 12.1).

LSB: база стандартов Linux

Free Standards Group — это некоммерческая организация, которая продвигает стандарты ПО с открытым исходным кодом и сейчас, в частности, занимается проектом по стандартизации Linux. На текущий момент в этом проекте принимает участие около 20 корпораций, среди которых и такие именитые гиганты-производители, как HP, IBM и Silicon Graphics. Главные дистрибьюторы Linux, такие как Red Hat, Novell (SUSE), Debian, Mandriva и Turbolinux, тоже принимают в нем участие. Цель проекта — упростить для сторонних разработчиков создание программных продуктов для Linux и сделать так, чтобы они становились частью одного целого, а не существовали по отдельности в виде 20 или 50 незначительно отличающихся версий.

В список подлежащих прохождению через процесс стандартизации элементов входят:

- интерфейсы библиотек, включая общие совместно используемые библиотеки;
- конфигурационные файлы;
- системные команды;
- общий формат пакетов и рекомендации по установке;
- API-интерфейс для системных интерфейсов;
- стандарты иерархий файловых систем.

Сейчас, на момент написания этой книги, самой последней версией является LSB 3.1, которая вышла в середине 2006 г. и включает LSB-ядро стандартов ISO (ISO/IEC 23360) и, впервые, настольные функциональные возможности, стандартизирующие инструментальные наборы для графических пользовательских интерфейсов Gtk и Qt. Китайское правительство, которое сейчас занимается разработкой стандарта для Linux в Китае, подписало соглашение о том, что в качестве основы для своего стандарта Linux будет использовать LSB.

Как ни печально, но пока что еще ни один производитель не выпустил никакого совместимого с LSB приложения. Получается, что эта замечательная идея, широко поддерживаемая в промышленных кругах, до сих пор не нашла своего применения.

POSIX

Комитет POSIX является подразделением организации IETF, которая взимает плату за копии своих стандартов, и последние несколько лет занимается разработкой единого стандарта для ОС UNIX. В прошлом он уделил немало внимания стандартизации команд, системных вызовов и библиотек функций. ОС Linux совместима со стандартами POSIX и следит за их развитием.

ITIL: библиотека интерфейсов информационных технологий

Стандарт ITIL (Information Technology Interface Library — библиотека интерфейсов информационных технологий) является скорее не техническим, а процедурным стан-

дартом для Linux, но при этом для крупных IT-организаций имеет наиболее важное значение, по сравнению с другими стандартами.

Система ITIL была разработана в Англии в 80-х гг. прошлого столетия для упрощения работы с мэйнфремами и разработки программного обеспечения в крупных организациях. Основной акцент делался на процессах, а не на людях для того, чтобы задачи внутри организации могли выполняться нормально независимо от того, уволился ли кто-то из сотрудников или ушел в отпуск. Были выделены двенадцать функциональных областей, которые довольно хорошо охватывали высокоуровневые IT-требования большинства европейских компаний (восстановление после аварий, управление изменениями, планирование нагрузки, и т.д.).

Система ITIL практически не применялась до тех пор, пока Internet не сделал информационные технологии обязательным компонентом большинства организаций. Крупным организациям стало необходимо как-то структурировать свои IT-процессы, а ITIL оказалась одним из наиболее подходящих для этого вариантов. В каждой из областей ITIL были выбраны наилучшие практические приемы.

Во избежание неразберихи каждое операционное IT-событие или действие в ITIL классифицируется как "изменение", "инцидент" (какое-то неожиданное событие) или "проблема" (событие, на исправление которого требуется определенное время).

Сейчас также появился стандарт ISO (ISO 20000), который похож на стандарт ITIL, так что время от времени можно встретить компании, которые утверждают, что их продукция является совместимой как с ITIL, так и с ISO. Обычно модель ITIL/ISO не совсем хорошо вписывается в среду организации и приходится нанимать консультативную фирму для ее настройки. Например, несколько из коммерческих диагностических систем, которые перечислялись в разделе 30.9, лишь частично соответствуют стандарту ITIL.

Многие крупные корпорации в США прибегают к стандарту ITIL для того, чтобы решить свои проблемы в области баз данных и системного администрирования, связанные с законом Сарбейнса-Оксли. Об этом законе и возникающих в связи с ним проблемах рассказывалось в разделе 30.12.

COBIT: задачи по управлению информационными и другими связанными с ними технологиями

Как и ITIL, COBIT (Control Objectives for Information and related Technology — задачи по управлению информационными и другими связанными с ними технологиями) представляет собой каркас для управления информацией и основывается на наилучших практических приемах. Целью COBIT является "исследование, разработка, публикация и распространение авторитетного, отвечающего современным нормам и пригодного для всех стран перечня общепринятых задач по управлению информационными технологиями, которым в своей повседневной работе могли бы пользоваться руководители и аудиторы".

COBIT является детищем Ассоциации аудита и контроля информационных систем (Information Systems Audit and Control Association, ISACA) и Института управления IT-технологиями (IT Governance Institute, ITGI). Подробности можно узнать по адресу `www.isaca.org/cobit`.

Первое издание каркаса COBIT было опубликовано в 1996 г., а на текущий момент он уже доступен в виде версии 4.0, которая была опубликована в 2005 г. На этот последний выпуск очень сильное влияние оказали требования закона SOX. Версия 4.0 со-

держит 34 высокоуровневых задачи, которые охватывают 215 “задач по управлению ”, разбитых на четыре раздела: “Планирование и организация”, “Приобретение и реализация”, “Доставка и поддержка” и “Мониторинг и оценка” (подождите-ка, а разве здесь не восемь разделов?).

30.15. Идеология Linux

В 80-е и 90-е гг. прошлого столетия пользователей ПК можно было отличить от пользователей Mac по их отношению к компьютеру. Первые просто считали компьютер полезным инструментом, а вторые любили его. У пользователей Mac компьютер был членом семьи, чем-то наподобие любимого домашнего животного.

Те же самые чувства, которые когда-то переполняли пользователей Macintosh, теперь испытывают пользователи Linux. Пользователи Linux не просто любят свои системы — они готовы сражаться за них, исправлять их, делать лучше, быстрее и безопаснее любой из систем Windows. Мир Linux наполняется жизненной силой с невероятной скоростью и в нем появляются свои этические нормы, свои мифы, свои боги и герои.

Группа энтузиастов Linux из города Берген в Норвегии, называющая себя BLUG (Bergen Linux User Group), заинтересовалась одним из первоапрельских документов серии RFC: RFC1149, *A Standard for the Transmission of IP Datagrams on Avian Carriers* (стандарт передачи дейтаграмм птичьей почтой). В этом документе описывался протокол CPIP (Carrier Pigeon Internet Protocol — Internet-протокол голубиной почты), и эти изобретательные норвежцы взяли и реализовали его при помощи стаи соседских голубей! Ниже приведен проверочный фрагмент их реализации (который был взят с Web-сайта проекта по адресу `www.blug.linux.no/rfc1149`):

```
Script started on Sat Apr 28 11:24:09 2001
vegard@gyversalen:~$ /sbin/ifconfig tun0
tun0   Link encap:Point-to-Point Protocol
       inet addr:10.0.3.2  P-t-P:10.0.3.1  Mask:255.255.255.255
       UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:150  Metric:1
       RX packets:1 errors:0 dropped:0 overruns:0 frame:0
       TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
       collisions:0
       RX bytes:88 (88.0 b) TX bytes:168 (168.0 b)

vegard@gyversalen:~$ ping -i 900 10.0.3.11[13]
PING 10.0.3.1 (10.0.3.1): 56 data bytes
64 bytes from 10.0.3.1: icmp_seq=0 ttl=255 time=6165731.1 ms
64 bytes from 10.0.3.1: icmp_seq=4 ttl=255 time=3211900.8 ms
64 bytes from 10.0.3.1: icmp_seq=2 ttl=255 time=5124922.8 ms

--- 10.0.3.1 ping statistics ---
9 packets transmitted, 4 packets received, 55% packet loss
round-trip min/avg/max = 3211900.8/5222806.6/6388671.9 ms
vegard@gyversalen:~$ exit
Script done on Sat Apr 28 14:14:28 2001
```

[13] Обратите внимание на использование частных адресов в соответствии с документом RFC1918. Понятно, что для соединения тестовой сети с внешним миром потребуется реализовать систему NAT на базе “голубиных” технологий.

Кто-то может сказать, что у этих ребят было слишком много времени в их распоряжении, но именно такой творческий подход и энтузиазм делают ПО с открытым исходным кодом (Linux в частности) таким мощным и интересным. Linux ликует!

30.16. Основные поставщики Linux

В настоящее время Linux является наиболее быстро развивающейся операционной системой и применяется везде, начиная с мэйнфреймов и заканчивая наручными часами. Однако она начинает потихоньку страдать от несовместимых компонентов и мелких отличий, которые в свое время стали преградой на пути развития UNIX.

Раньше в корпоративных кругах нередко можно было услышать: "Что вы! Мы не можем использовать ОС Linux, ведь она не поддерживается". Но ситуация меняется так быстро, что компания Microsoft уже не может не беспокоиться. Linux стремительно прокладывает себе путь в мир бизнеса. Часто носителями новых идей являются только что нанятые на работу студенты, которые работали с Linux в университете и продолжают использовать ее в качестве настольной системы. Linux становится все более удобной платформой для почтовых и Web-серверов, где благодаря надежности, производительности и масштабируемости этой операционной системы ее предпочитают коммерческим продуктам компании Microsoft. Стоимость владения также является одним из главных преимуществ Linux.

Слово "бесплатное" в выражении "бесплатное программное обеспечение" имеет два значения: первое — как у слова в выражении "бесплатное пиво", подразумевающем, что за это программного обеспечение не нужно платить, а второе -— как у слова "свободный", подразумевающем наличие относительно небольшого количества ограничений в том, что с этим программным обеспечением разрешается делать. Лицензия GPL (GNU Public License — открытая лицензия GNU) требует применения обоих этих значений, и этот факт пугает некоторые IT-отделы. Средства, разработанные на основе GPL-продуктов, в случае распространения обязательно тоже должны отвечать требованиям этой лицензии. Некоторые руководители воспринимают разработчиков Linux как длинноволосых хиппи и относятся к GPL с сомнением, неуверенностью и страхом. Но, похоже, Linux медленно, но уверенно завоевывает и их сердца.

В течение последних нескольких лет на сайте `www.distrowatch.com` проводится голосование с целью выявления 100 наилучших дистрибутивов Linux; места распределяются в соответствии с количеством полученных от посещающих этот сайт пользователей голосов. Места пяти рассматриваемых нами дистрибутивов постоянно меняются, как видно в табл. 30.3.

Таблица 30.3. Рейтинг популярности дистрибутивов Linux на сайте Distrowatch.com

Дистрибутив	Последние полгода	Последний год	2005 год	2004 год	2003 год
Ubuntu	1	1	1	13	-
SUSE	2	2	3	4	7
Fedora	3	4	4	2	-
Debian	7	7	6	5	5
Red Hat	29	24	23	12	2

Количество поступающих за день голосов варьируется от 100 до 3 000. Эти рейтинги как ничто другое демонстрируют рост популярности Ubuntu и разрыв между Red Hat Enterprise Linux и Fedora.

В настоящее время все крупные компании занимаются выпуском и поддержкой Linux-систем, среди них IBM, Hewlett-Packard, Silicon Graphics, Sun, и др. Интересно было наблюдать за сближением этих огромных, медлительных в принятии нового корпораций с миром Linux, где всем заправляют молодые неопытные инженеры-программисты, чья энергия с лихвой компенсирует все их "острые углы". Корпорации IBM, например, пришлось серьезно изменить свой подход к выпуску программного обеспечения, чтобы приспособиться к анархизму открытой среды Linux. Она внесла свой вклад в разработку открытых систем, перенеся в Linux свои многочисленные библиотеки хорошо отлаженного кода. Подробнее об этом можно узнать по адресу `ibm.com/linux` или `ibm.com/developerworks/opensource`.

Компания SGI организовала проект "Linux Test Project" (Проект тестирования Linux), заключающийся в разработке ряда тестовых пакетов и методов нагруженного тестирования для проверки надежности, устойчивости и стабильности Linux. Теперь этим проектом занимается корпорация IBM. Самая последняя его версия (на июнь 2006 г.) включает свыше 2 900 тестов плюс документацию и средства тестирования.

Проект LDP (`www.tldp.org`) — Linux Documentation Project (Проект по обеспечению документации для Linux) — подразумевает выпуск различной документации по Linux вроде подробных ознакомительных буклетов, справочников по наиболее часто задаваемым вопросам, руководств по выполнению различных, наиболее распространенных задач и `man`-страниц по всем базовым командам и пакетам Linux[14]. В рамках проекта LDP также выпускаются два электронных журнала: *Linux Gazette* и *Linux Focus*. Вся документация доступна бесплатно в сети Internet.

По крайней мере, одно из страховых агентств, работающих в сфере компьютерной безопасности, учитывает степень устойчивости Linux в своих расценках и берет за страхование Web-серверов на базе Linux на 5–15% процентов меньше, чем за страхование аналогичных систем Microsoft. В августе 2001 г. компания ZDNet опубликовала статью, в которой содержались такие комментарии[15]:

"Страховое агентство J.S. Wurzler Underwriting Managers после атаки вируса Code Red на 15% увеличило страховые взносы с клиентов, которые пользуются программой Internet Information Server производства компании Microsoft. Агентство, занимающееся страхованием от хакеров с 1998 г., обосновало свое решение результатами более чем 400 аналитических исследований, проведенных за последние три года. Было обнаружено, что системные администраторы, имеющие дело с открытыми системами, лучше подготовлены и реже меняют место работы, чем администраторы компаний, где используется программное обеспечение Windows".

Из пяти рассматриваемых нами дистрибутивов крупными коммерческими организациями поддерживаются только два: это дистрибутив Red Hat, поддерживаемый корпорацией Red Hat, и дистрибутив SUSE, поддерживаемый компанией Novell. Дистрибутив

[14] Раньше официальный Web-сайт этого проекта назывался `linuxdoc.org`, но внутренний скандал, разразившийся из-за того, что Web-мастер по забывчивости или намеренно позволил взять имя такое же как у коммерческого Web-сайта `linuxdoc.com`, вынудил руководителей этого проекта сменить домен.

[15] Полную версию этой статьи можно найти по адресу `www.zdnet.com/zdnn/stories/news/0,4586,2805929,00.html`.

Ubuntu поддерживается небольшой компанией под названием Canonical Ltd, которая была основана уроженцем Южной Африки Марком Шатлвортом (Mark Shuttleworth). Дистрибутивы Debian и Fedora пока что поддерживаются совсем небольшими группами энтузиастов.

В сообществе Linux между разработчиками Debian и разработчиками ориентированных больше на предприятия дистрибутивов, таких как Red Hat и SUSE, произошел своего рода раскол, в результате которого теперь даже хорошие идеи первых никто не хочет воспринимать всерьез и включать в LSB-проекты по стандартизации. Мы надеемся, что ставший недавно популярным дистрибутив Ubuntu поможет Debian более надежно закрепиться среди основных дистрибутивов Linux.

Дистрибутив CentOS (Community ENTerprise Operating System), разрабатываемый в Англии, начинает пользоваться все большей популярностью у тех, кто хочет получить стабильную систему Linux без участия Red Hat. Его в принципе можно назвать клоном Red Hat Enterprise Linux. Согласно условиям лицензии GPL компания Red Hat обязана делать свои пакеты программ доступными для общественности. Этим и пользуются разработчики CentOS: они обрабатывают деревья исходного кода, удаляют из них логотипы и товарные знаки Red Hat, а затем воссоздают дистрибутив с нуля, выпуская подобную RHEL, но бесплатную и не требующую подписания контрактов на поддержку версию. Конечно, поддержка играет далеко не самую последнюю роль, а компания Red Hat обещает поддерживать RHEL на протяжении семи лет, что в мире Linux эквивалентно бесконечности.

Для большинства компаний общая стоимость владения системой тоже очень важна. Приобретение почти всех дистрибутивов обходится бесплатно, а благодаря встроенным в них средствам сетевого управления, их обслуживание также обходится очень дешево. Например, стоимость владения Windows XP Professional в настоящее время составляет 250 долларов за единицу, причем столько стоит только сама операционная система. В случае добавления в эту операционную систему дополнительных приложений стоимость ее владения в любой крупной компании может достигать действительно устрашающих цифр. В корпорации Google.com сейчас используются сотни тысяч компьютеров, на которых установлены различные дистрибутивы Linux: а теперь попробуйте представить, во сколько бы обошлось этой корпорации приобретение лицензий Windows для всех ее компьютеров.

Программное пиратство просто-таки процветает на Дальнем Востоке. Желая избавиться от него, китайское правительство решило отныне пропагандировать операционную систему Linux, а не Windows, потому что она распространяется с открытым исходным кодом и потому непригодна для пиратства. Дистрибутивов, поддерживающих китайский язык, существует пока только несколько. Red Flag Linux является одним из самых популярных среди них и насчитывает свыше 200 миллионов пользователей.

Linux используется во многих правительственных учреждениях США, таких как NASA, Национальный институт здоровья, Национальное управление по исследованию океанов и атмосферы, Служба геологии, геодезии и картографии США, а также Национальные лаборатории Fermilab в Лос-Аламосе, Окридже и Сандии. Многие из наиболее мощных компьютеров сегодня по-прежнему представляют собой “большие железяки”, но на смену им приходят кластеры Beowulf (`www.beowulf.org`), состоящие из компьютеров Linux. В сентябре 2004 г. корпорация IBM объявила о том, что создала самый быстрый суперкомпьютер в мире, а именно — кластер из 16 250 ЦП, уместившихся всего в восьми стойках и функционирующих под управлением Linux с постоян-

ной производительностью в 36 терафлопс (триллионов вычислений с плавающей запятой в секунду).

30.17. Организации, конференции и другие ресурсы

В мире существует множество групп поддержки UNIX и Linux — как систем в целом, так и отдельных их версий, которые помогают пользователям налаживать контакты с другими пользующимися аналогичным ПО людьми. В табл. 30.4 перечислены некоторые, наиболее известные из этих групп. Но существует также множество групп и организаций национального и регионального уровня, которые в эту таблицу включены не были.

Таблица 30.4. Группы поддержки Linux и UNIX

Название	Адрес	Комментарии
LPI	www.lpi.org	Институт профессионалов Linux
LI	www.li.org	Международная организация пользователей Linux
FSF	www.fsf.org	Фонд бесплатного ПО, является спонсором проекта GNU
OSDL	www. osdl.org	Лаборатория по разработке ПО с открытым исходным кодом; подходит для разработчиков
USENIX	www.usenix.org	Группа пользователей UNIX, занимающаяся техническими вопросами
SAGE	www.sage.org	Гильдия системных администраторов, связанная с USENIX; каждый год проводит конференцию LISA
LOPSA	www.lopsa.org	Союз профессиональных системных администраторов; эта организация появилась в результате раскола между USENIX и SAGE
SANS	www.sans.org	Проводит конференции по вопросам системного администрирования и безопасности; в отличие от SAGE, больше ориентирована на обучение администраторов
EUROPEN	www.europen.org	Некогда представляла собой объединенный союз национальных групп пользователей, но сейчас уже практически не функционирует; остатки по-прежнему работают в NLUUG, DUUG, UKUUG, GUUG и других подобных организациях
AUUG	www.auug.org.au	Австралийская группа пользователей UNIX; занимается как техническими, так и связанными с управлением аспектами обработки данных
SAGE-AU	www.sage-au.org.au	Австралийское отделение организации SAGE; каждый год проводит конференции в Австралии
SANE	www.sane.nl	Группа пользователей, занимающаяся вопросами системного администрирования и сетевых технологий; каждый год проводит конференции в северной Европе

Некоторые из этих организаций и групп поддержки проводят конференции на различные, посвященные вопросам Linux темы, на которых также очень часто рассматриваются не только вопросы Linux и UNIX, но и события и тенденции в мире Windows.

Организация Linux International (LI) и Институт профессионалов Linux (Linux Professional Institute, LPI) занимаются продвижением Linux: LI — путем проведения маркетинговых исследований в области коммерческого использования Linux (и спонсирования некоторых проектов по разработке с открытым исходным кодом), а LPI — посредством своей программы сертификации системных администраторов, о которой более подробно будет рассказываться в следующем разделе.

Web-сайт организации LI является неплохой "приманкой" для руководителей, которые очень скептически относятся к использованию операционных систем с открытым исходным кодом. А сама она спонсирует продукты Linux на нескольких крупных торговых выставках, включая LinuxWorld, LinuxExpo и CeBit. Она также ведет проект интернационализации, целью которого является перевод всех существующих наборов символов в Linux. В рамках программы летней интернатуры организация дает возможность студентам пообщаться с разработчиками Linux и приобрести настоящий профессиональный опыт в написании качественного кода.

Организация Linux International сейчас уже начала потихоньку сворачивать свою деятельность, потому что цель, для которой она создавалась изначально (а это — внедрение Linux в мир корпораций), достигнута. Не исключено, что в ближайшем будущем она превратиться в какую-нибудь обычную группу пользователей Linux.

Организация FSF (Free Software Foundation — Фонд бесплатного ПО) является спонсором проекта GNU (GNU — это рекурсивное сокращение от "GNU is Not Unix"; так называется проект по свободному распространению программного обеспечения). Под словосочетанием "бесплатное ПО" в названии этой организации подразумевается ПО, которое не имеет почти никаких ограничений в использовании, а не ПО, за которое не нужно платить. Также FSF является автором лицензии GPL, которая распространяется на большую часть продуктов Linux. В настоящее время готовится следующая версия этой лицензии, которая будет называться "GPLv3"; ее черновые разделы уже сейчас доступны для комментариев общественности.

Организация OSDL (Open Source Development Lab — Лаборатория по разработке ПО с открытым кодом) представляет собой благотворительную организацию, целью которой является ускорить распространение Linux в корпоративной среде. OSDL поддерживается основными заказчиками Linux и предоставляет тестовую и техническую поддержку для разработчиков Linux.

Конференции и торговые выставки

В табл. 30.5 представлены детали трех проводимых ежегодно конференций по вопросам разработки Linux.

Таблица 30.5. Конференции по вопросам разработки Linux

Конференция	URL-адрес	Длительность	Страна
Linux Symposium (Симпозиум по вопросам разработки Linux)	`www.linuxsymposium.org`	4 дня	Оттава, Канада
linux.conf.au	`linux.conf.au`	5 дней	Австралия
Linux Kongress (Конгресс по вопросам разработки Linux)	`www.linux-kongress.org`	4 дня	Германия

Конференция в Оттаве обычно проводится в августе и состоит из трех этапов, во время которых зачитываются различные доклады и параллельно проводятся различные консультации. Австралийская конференция подразумевает проведение практических занятий, семинаров, зачитывание основных докладов и проведение мини-конференций, одна из которых всегда обязательно посвящена теме системного администрирования. Эта конференция обычно проходит в январе в разных городах Австралии. Немецкая конференция — Linux Kongress (Конгресс по вопросам разработки Linux) — включает два дня практических занятий и два дня технических презентаций. Место проведения постоянно меняется, что касается времени, то чаще всего она проводится в сентябре.

Linux Weekly News (`lwn.net`) — это электронный журнал, посвященный Linux. Его в 1997 г. создали Элизабет Кулбаух (Elizabeth Coolbaugh) и Джонатан Корбет (Jonathan Corbet). На него можно подписаться; подписавшиеся пользователи сразу же получают доступ ко всем статьям, а тем, кто не подписался, приходится ждать около недели. Стоимость подписки варьируется и зависит от категории, к которой относится подписчик. "Жаждущим знаний хакерам" (категория "Starving hacker") журнал обходится дешевле всего, а именно — 2 доллара 50 центов в месяц; "профессиональным хакерам" (категория "Professional hacker") — 5 долларов в месяц, а "руководителям проекта" (категория "Project Leader") — 10 долларов в месяц.

Организация USENIX, представляющая собой союз пользователей Linux, UNIX и других операционных систем с открытым исходным кодом, каждый год проводит одну общую конференцию и несколько специализированных (небольших) конференций. На общей конференции обычно речь идет об открытых системах и новых разработках, которые проводятся в рамках Linux и BSD.

Важным событием для системных администраторов является конференция LISA (Large Installation System Administration — системного администрирования в крупных организациях), которая проводится поздней осенью. На фоне всех этих конференций часто проводятся соответствующие торговые выставки.

В течение нескольких последних лет организация USENIX также специально для сообщества Linux проводит отдельную конференцию по вопросам разработки ядра Linux. Попасть на это двухдневное мероприятие можно только по приглашению.

SAGE (USENIX System Administration Guild — гильдия системных администраторов USENIX) является первой международной организацией для системных администраторов. Она рекламирует системное администрирование как профессию путем финансирования различных конференций и неофициальных программ. Подробнее о ней можно узнать на сайте `www.sage.org`.

В 2005 г. раскол между USENIX и SAGE поставил будущее SAGE под сомнение. В результате этого раскола появилась новая отдельная организация под названием "LOPSA" (League of Professional System Administrators — Союз профессиональных системных администраторов), основанная бывшими сотрудниками организации SAGE. Он пока не проводит никаких конференций, но не исключено, что очень скоро все-таки начнет это делать. У SAGE была программа по сертификации системных администраторов, но она от нее отказалась; будем надеяться, что LOPSA вернется к ней.

Во многих областях существуют региональные группы пользователей UNIX, Linux и других открытых систем. Одни из них являются филиалами организации USENIX, другие — нет. Местные группы обычно организовывают регулярные встречи и обучение с участием как местных, так и приезжих специалистов, а также нередко обедают вместе перед или после этих встреч. Они являются прекрасным способом наладить контакты с другими системными администраторами в данном регионе.

Крупнейшей торговой выставкой-ярмаркой в мире Linux является выставка LinuxWorld Expo, которая в настоящее время проводится раз в год в США, обычно в городе Сан-Франциско и, как правило, осенью. Кроме того, она также целый год повторно проводится в различных городах Европы и Азии; вскорости планируется проводить ее и в таких крупных городах, как Пекин, Москва и Сингапур.

Главной торговой выставкой в сфере сетевой индустрии является выставка Interop. Проводимые в ее рамках обучающие семинары тоже славятся высоким качеством. Когда-то выставка Interop проходила только раз в год; этого события с одинаковым нетерпением ждали как технические специалисты, так и производители. Сейчас она проводится несколько раз в год, можно сказать, представляя собой "бродячий цирк сетевых технологий". Зарплата преподавателей семинаров сократилась вполовину, но качество семинаров вроде бы от этого не пострадало.

LPI: институт профессионалов Linux

Организация LPI занимается сертификацией системных администраторов Linux. Она разрабатывает тесты, позволяющие оценить уровень знаний администраторов и их умение решать различные задачи. Существует три уровня сертификации, на каждом из которых требуется сдать тест, состоящий из двух частей. Вопросы тестов подразумевают выбор одного из нескольких вариантов ответа или предоставление краткого ответа. Имеющиеся на сегодняшний день тесты (для уровней 1 и 2) доступны в Internet на сайте компании Vue Electronic Testing Service (`www.vue.com/lpi`).

На первом уровне сертификации охватываются команды привилегированных пользователей Linux и основные административные задачи. Уровень 2 — более углубленный и охватывает сетевое администрирование. Уровень 3 определен, но еще не полностью реализован. Как ни печально, но состояние уровня 3 ничуть не изменилось с момента выхода предыдущего издания этой книги, который состоялся уже целых пять лет назад. На Web-сайте LPI (`www.lpi.org`) объясняется, какой объем знаний ожидается на каждом уровне сертификации, и предлагаются образцы вопросов по каждому из тестов.

Сертификация, несомненно, важна, но добиться положительных результатов не так-то просто. Системные администраторы обладают, прежде всего, практическими навыками. Тесты, в которых предлагаются варианты ответов, легко составлять и оценивать, но они не очень хорошо отражают практические навыки администрирования. Мы надеемся, что тесты уровня 3 будут более практическими, больше напоминающими экзамен Cisco CCIE, а не Microsoft MCSE.

У Red Hat и SUSE есть свои собственные программы сертификации. Программа сертификации по Ubuntu на текущий момент в основном дублирует программу сертификации LPI.

Списки рассылки и Web-ресурсы

Сегодня для системных администраторов доступно просто огромное количество самых различных списков рассылки и Web-ресурсов. Некоторые из них мы перечисляли в главе 1. Но помимо этого, очень обширный перечень списков рассылки, посвященных Linux, также доступен по адресу `www.linux.org/docs/lists.html`.

Ответы на вопросы по Linux очень эффективно можно искать при помощи соответствующего поискового механизма Google: просто зайдите на `www.google.com.linux` и посмотрите, насколько хорошо работает этот механизм.

Также существуют списки рассылки, посвященные конкретным дистрибутивам; в табл. 30.6 приведены несколько адресов списков рассылки, посвященных рассматриваемым в этой книге дистрибутивам.

Таблица 30.6. Адреса списков рассылки для различных дистрибутивов

Дистрибутив	URL-адрес
Red Hat	`www.redhat.com/mailing-lists`
Fedora	`www.redhat.com/mailing-lists`
SUSE	`www.suse.com/us/support/online_help/mailinglists/`
Debian	`www.debian.org/MailingLists/`
Ubuntu	`lists.ubuntu.com`

Для Red Hat доступно несметное количество списков рассылки, которые охватывают как дистрибутивы Enterprise Linux, так и дистрибутивы Fedora; на странице `www.redhat.com/mailing-lists` приведен полный перечень этих списков с их названиями и кратким описанием.

Помимо URL-адреса, указанного в таблице, информацию о списках рассылки для SUSE также можно найти и на странице `www.suse.com/us/business/mailinglists.html`. Архивы же этих списков рассылки доступны по адресу `lists.suse.com/archives`.

Опросы системных администраторов

Организация SAGE (System Administrators Guild — Гильдия системных администраторов) совместно с ассоциацией USENIX и Институтом системного и сетевого администрирования и проблем безопасности (System Administration, Networking, and Security Institute, SANS) каждый год проводят опросы, посвященные в основном работе администраторов, но также охватывающие и некоторые другие темы. Эти опросы можно найти на сайте `www.sage.org/salsurv` и `www.sans.org/salary2005`. Последние опросы SAGE (занимающие в целом приблизительно 50–60 страниц) доступны только для зарегистрированных пользователей; старые опросы может загрузить каждый. На Web-сайте SANS можно просмотреть только краткий, занимающий 5 страниц в формате PDF отчет о результатах самого последнего опроса.

Чаще всего посредством этих опросов собираются следующие статистические данные.

- сведения о зарплате: средняя зарплата, хронология повышения зарплаты, бонусы (премии) и сверхурочные;
- комбинация обслуживаемых платформ;
- количество рабочих часов в неделю;
- спрос на системных администраторов на рынке труда;
- объем опыта, уровень знаний, число проведенных на данной работе лет;
- место проживания, пол и возраст администраторов;
- самая нелюбимая и самая любимая части работы.

Интересно будет сравнить статистические данные своей организации с аналогичными данными других организаций в мире по результатам этих опросов. Однако к данным

опросов следует относиться с осторожностью. На результаты всегда влияют местные условия и состояние экономики. Зарплаты, прожиточный минимум и налоги могут очень сильно отличаться в разных странах и даже разных областях.

30.18. Рекомендуемая литература

Инфраструктура

- Tragott, Steve. *Bootstrapping and Infrastructure*. Boston: LISA 1998. `www.infrastructures.org/papers/bootstrap/bootstrap.html`
- Scheikert, David. *ISGTC: an alternative to ~bofh/bin*. Amsterdam, Netherlands: SANE 2004. `isg.ee.ethz.ch/publications/papers/isgtc-sane.pdf`
- Oetiker, Tobias. *TemplateTree II: The Post-Installation Setup Tool*. San Diego, CA: LISA 2001. `isg.ee.ethz.ch/tools/tetre2/pub/tetre-lisa.pdf`
- Burges, Mark S. *Principles of Network and System Administration. 2nd Edition*. Hoboken, NJ: Wiley, 2003.
- Многие из средств и концепций, которые описывались в посвященном управлению разделе этой главы, можно найти на сайте `isg.ee.ethz.ch/tools`.

Управление

- Limincelli, Thomas A. and Christine Hogan. *The Practice of System and Network Administration*. Boston, MA: Addison-Wesley, 2001.
- У этой книги есть свой Web-сайт, на котором также доступно много интересной информации: `www.everythingsysadmin.com`. Следующее издание книги ожидается в 2007 г.
- Machiavelli, Niccolo. *The Prince. 1513*. Доступна в электронном виде на сайте `www.gutenberg.org/etext/1232/`.
- Limoncelli, Thomas A. *Time Management for System Administrators*. Sebastopol, CA: O'Reilly Media, 2005.
- Brooks, Frederick P., Jr. *The Mythical Man-Month: Essays on Software Engineering*. Reading, MA: Addison-Wesley, 1995.
- Замечательная книжка небольшого размера; была впервые опубликована в 1975 г., но остается актуальной и по сей день.
- McConnell, Steve. *Software Project Survival Guide: How to Be Sure Your First Important Project Isn't Your Last* . Redmond, WA: Microsoft Press, 1998.
- Эта книга написана в непринужденной манере и содержит много полезной информации.
- Pressman, Roger S. *Software Engineering: A Practitioner's Approach (6th Edition)*. Boston, MA: McGraw-Hill, 2005.
- Somerville, Ian. *Software Engineering (8th Edition)*. New York, NY: Addison-Wesley, 2006.
- При желании ознакомится с жаргонизмами, ассоциируемыми с процессами и стандартами ITIL, лучше всего начать с сайта `www.itil-toolkit.com`.

Политика и безопасность

- *Site Security Handbook* (Руководство по обеспечению безопасности сайта) версия RFC2196 и версия RFC1244. Содержимое отличается значительно, поэтому стоит просмотреть обе версии.
- Локальные политики, стандарты и процедуры суперкомпьютерного центра (Supercomputer Center) в Сан-Диего доступны на сайте `security.sdsc.edu/help/SGs.shtml`. На этом сайте представлена замечательная коллекция программных документов, но, к сожалению, он уже пару лет не обновлялся.

Правовые вопросы, патенты и конфиденциальность

- Web-сайт организации Electronic Frontier Foundation (`eff.org`) — это место, где можно найти самые последние комментарии по вопросам защиты персональной информации, шифровании и законах. Его всегда интересно читать.
- `www.groklaw.net` — это замечательный сайт, на котором всегда можно найти самые последние новости по юридическим вопросам, касающимся информационных технологий. Вся информация излагается и суммируется на доступном языке, а не на языке юридических терминов, понятном только для узкого круга людей.

Общие индустриальные новости

- `slashdot.org` — прекрасный источник новостей в сфере технологий, но часто на день или два "отстает" от других источников, таких как посвященные теме безопасности списки рассылки.
- Harrow, Jeffrey R. The Harrow Technology Report. `www.theharrowgroup.com`.

  На этом сайте доступны некоторые интересные статьи на тему развития технологий. Здесь есть и новости, и обучающий материал. Джефри Хэрроу (Jeffrey R. Harrow) является автором периодического издания *The Rapidly Changing Face of Computing* (Стремительно изменяющийся мир вычислительных технологий), которое ранее публиковалось издательством Compaq, но уже больше не выходит. Web-сайт обновлялся до конца 2005 г.
- `www.heise.de` — прекрасный источник новостей. Он на немецком языке, но на странице `www.heise.de/english` дается английский перевод.
- `www.theregister.co.uk` или `www.theregister.com` — хороший сайт новостей в сфере информационных технологий.

30.19. Упражнения

30.1. Какие процедуры в вашей организации приходится выполнять регулярно. Какие из них доводится выполнять редко, но каждый раз с новым подходом? Какие из них являются рискованными?

30.2. В чем проявляется зависимость от внешних поставщиков? Нужен и имеется ли план Б? Объясните причины его наличия или отсутствия. Если план Б существует, опишите его.

30.3. Проведите небольшой опрос внутренних пользователей, чтобы определить их ожидания в отношении существующей вычислительной инфраструктуры. Совпадают ли их ожидания? Являются ли они разумными? Совпадают ли они с задачами, определенными группой администраторов?

30.4. Какая организованная инфраструктура для системного управления уже установлена? Каких компонентов не хватает?

30.5. Один из сотрудников собирается уйти завтра на обед и больше никогда не вернуться, но какой именно — пока не известно. На какие критические процедуры это может повлиять и насколько организация готова к замене отсутствующего сотрудника? Какая документация должна быть в наличии, чтобы можно было избежать сбоя в предоставлении услуг?

30.6. Что произойдет, если *вы*, администратор, исчезнете на три месяца? Насколько сильно вас будут ненавидеть коллеги, когда вы, наконец, вернетесь, и почему? Что можно сделать за две недели, чтобы сократить негативные последствия такого случая?

☆ 30.7. Начальство приказало урезать выделяемый на системное администрирование бюджет на 30% к концу текущего года. Можете ли вы оценить последствия такого урезания? Составьте краткий отчет, который позволит начальнику принять осмысленное решение по поводу того, какие службы следует сократить или убрать.

☆ 30.8. Какие крупные корпорации в настоящий момент активно поддерживают Linux? В чем их заинтересованность? Какой вклад они вносят?

☆ 30.9. Вы восстанавливаете систему после сбоя диска и обнаруживаете файлы в каталоге `lost+found`. При дальнейшем анализе выясняется, что некоторые из файлов являются почтовыми сообщениями, которые посылали друг другу два студента, пытающиеся в обход брандмауэра организовать хранение MP3-файлов на удаленном файловом сервере. Каковы будут ваши действия? Существуют ли в организации документы, предусматривающие правила поведения системного администратора при таких нарушениях?

☆☆ 30.10. Оцените полноту имеющейся в вашей организации локальной документации для новых пользователей, системных администраторов, стандартных процедур и аварийных ситуаций.

☆☆ 30.11. Составьте прогноз развития различных коммерческих и бесплатных версий UNIX и Linux на ближайшие пять лет. Как видоизменятся текущие проекты и модели поставки дистрибутивов? Какое влияние окажет поддержка Linux со стороны производителей аппаратных средств? Проведите различия между рынком серверов и рынком настольных систем.

Предметный указатель

A

ACL (access control system), 132
ADSM, 246
API (Application Programming Interface), 114
APT, 296
ARP (Address Resolution Protocol), 330; 350
ATM (Asynchronous Transfer Mode), 420

B

Bacula, 228
BGP (Border Gateway Protocol), 394
BIND, пакет
  аппаратные требования, 480
  версия, 468; 483
  дистрибутив ISC, 468
  журнальная регистрация, 525
    каналы, 527
    категории сообщений, 528
  клиент, 478
  компоненты, 470
  конфигурационные файлы, 480
  конфигурация, 498; 500
  параметры конфигурации, 482
    allow-query, 487
    allow-recursion, 484
    allow-transfer, 487
    also-notify, 484
    blackhole, 487
    directory, 483
    files, 485
    forward, 487
    forwarders, 486
    listen-on, 485
    notify, 484
    query-source, 485
    recursion, 484
    rrset-order, 487
    sortlist, 487
    topology, 487
    transfer-format, 484
    transfer-source, 485
    transfers-in, 485
    transfers-out, 485
    transfers-per-ns, 485
    version, 483
  сервер, 480
  сообщения об ошибках, 529
  статистика, 532
BIOS (Basic Input/Output System), 66
BRI (Basic Rate Interface), 422

C

CGI-сценарий, 796
CIDR (Classless Inter-Domain Routing), 337
CIFS (Common Internet File System), 907
CUPS (Common UNIX Printing System), 976

D

DCE (Data Communications Equipment), 925
DDS (Digital Data Service), 421
Debian
  инсталляция пакетов, 289; 293
DHCP (Dynamic Host Configuration Protocol), 982
DNS, 431; 434; 436; 437
  база данных
    директива $ORIGIN, 451
    директива $TTL, 448; 453
    записи о ресурсах, 448
    запись A, 454; 466
    запись CNAME, 440; 458
    запись LOC, 460
    запись MX, 456
    запись NS, 454; 466
    запись NXT, 519
    запись PTR, 455; 464
    запись SIG, 519
    запись SOA, 433; 451
    запись SRV, 461; 523
    запись TXT, 462
    запись WKS, 462
    класс записи, 449
    обновление, 506; 508
    связующие записи, 466
  делегирование, 442
  динамические обновления, 493
  домен
    выбор имени, 441
    географический, 437
    организационный, 437
    принципы управления, 440
    регистрация, 441
    создание поддоменов, 442
  зона, 451
    порядковый номер, 433; 452
    создание, 492
    цифровая подпись, 516
  зонные пересылки, 471; 472; 484; 507
    инкрементные, 507
  кэширование, 443
  напрасное делегирование, 442; 534
  обратное преобразование, 455
    в стандарте IPv6, 463
    посредством записей CNAME, 458

прямое преобразование, 454
распознаватель, 436; 470; 473
конфигурирование, 478
тестирование, 479
расщепление, 497
сервер имен, 436; 470; 471
авторитетный, 454; 472
главный, 472; 492
кэширующий, 472
невидимый, 454
нерекурсивный, 472
переадресующий, 486; 495
подчиненный, 472; 493
рекурсивный, 472
усеченный, 467; 472; 493
сигнатуры транзакций, 513
спецификация
DNSSEC, 446; 490; 516
EDNS0, 444; 491
GSS-TSIG, 523
TKEY, 513
TSIG, 446; 513
файл подсказок, 473; 492; 494
DPMS (Display Power Management Signaling), 825
DSL (Digital Subscriber Line), 422
DTE (Data Terminal Equipment), 925
DVMRP (Distance Vector Multicast Routing Protocol), 398

E

Eiiaiaa
makemap, 658
Ethernet, 408
EULA (End User License Agreements), 1040

F

FDDI (Fiber Distributed Data Interface), 418
FIFO (First In, First Out), 124
Frame Relay, 421

G

GECOS (General Electric Comprehensive Operating System), 142
GNOME (GNU Network Object Model Environment), 833

I

ICANN (Internet Corporation for Assigned Names and Numbers), 327
ICMP (Internet Control Message Protocol), 330
IDE (Integrated Drive Electronics), 157
IETF (Internet Engineering Task Force), 327
Information Technology Interface Library, 987
IP (Internet Protocol), 329
ISDN (Integrated Services Digital Network), 422
ISOC (Internet Society), 327

K

KDE (K Desktop Environment), 832

L

LDAP (Lightweight Directory Access Protocol), 911
Linux, 43; 61
безопасность, 742; 749; 758
дистрибутивы
защищенные, 784
инсталляция, 276
проект LSB, 1047
LVM (logical volume management), 186
LVM (Logical Volume Manager), 187

M

MS DES (Microsoft's Distributed File System), 913

N

NAT, 780
Netfilter, 779
NFS (Network File System), 93; 201; 543
безопасность, 546; 760
блокировка файлов, 545
версии, 544
выбор порта, 555
выбор транспортного протокола, 544
дисковые квоты, 545
имена файловых систем, 546
клиент, 552
пользователь
nobody, 547
root, 547
секретный ключ, 545
сервер, 548
специализированный, 556
Niai, 664
NIS (Network Information Service), 573; 574; 576; 577; 985
безопасность, 759
демоны, 576
домен, 573
создание, 580
карта, 573
ypservers, 575
команды, 576
конфигурирование
клиентов, 582
сервера, 580
сетевые группы, 579

O

OTRS (Open Ticket Request System), 1021

P

PAM (Pluggable Authentication Modules), 565; 754; 817
PAT (Port Address Translation), 374
PCL (Printer Control Language), 838
PDF (Portable Document Format), 838
PIM (Protocol Independent Multicast), 398
PJL (Printer Job Language), 839
PP (Internet Printing Protocol), 976
PPP (Point-to-Point Protocol), 375
PXE (Pre-boot eXecution Environment), 982

R

RAID (Redundant Array of Independent Disks), 185
RDP (Remote Desktop Protocol), 903
Red Hat
  инсталляция, 279
Red Hat Enterprise Linux, 46
Red Hat Network, 292
RIP (Routing Information Protocol), 393
RPC, 544
RPM, 287; 288

S

Samba, 907
SCSI (Small Computer Systems Interface), 157
SEAL (Simple and Efficient Adaptation Layer), 420
sendmail, почтовая система, 596; 622
  база доступа, 655; 658; 661
  безопасность, 670; 760
  библиотека libmilter, 655; 664
  версии, 594; 623
  группы очередей, 629; 679
  журнальная регистрация, 687
  изменение корневого каталога, 675
  инсталляция, 624
    в Debian, 626
  коды состояния доставки, 659
  конфигурирование, 633; 634; 635; 640
    доменный файл, 634; 638
    опции, 652
  макроконстанта
    LUSER_RELAY, 658
    MAIL_HUB, 649; 667
    SMART_HOST, 649; 667; 668
  макросы, 632
    CANONIFY_DOMAIN, 642
    CANONIFY_DOMAIN_FILE, 642
    DOMAIN, 638
    EXPOSED_USER, 648
    FEATURE, 640
    GENERICS_DOMAIN, 645
    GENERICS_DOMAIN_FILE, 645
    LDAPROUTE_EQUIVALENT, 647
    LDAPROUTE_EQUIVALENT_FILE, 647
    MAILER, 638
    MASQUERADE_AS, 647
    MASQUERADE_DOMAIN, 648
    MASQUERADE_DOMAIN_FILE, 648
    MASQUERADE_EXCEPTION, 648
    MASQUERADE_EXCEPTION_FILE, 648
    OSTYPE, 636
    RELAY_DOMAIN, 657
    RELAY_DOMAIN_FILE, 657
    VERSIONID, 636
    VIRTUSER_DOMAIN, 646
    VIRTUSER_DOMAIN_FILE, 646
  маскирование адресов, 647; 649
  направление почты в программу, 612; 673
  направление почты в файл, 611; 673
  обработка недоставленных сообщений, 681
  опции безопаснопое, 674
  отладка, 683
  очередь сообщений, 629
    обработка, 680
  переадресация почты, 614; 641
  права доступа, 672
  препроцессор m4, 631; 632
  проверка заголовков, 661
  производительность, 681
  псевдонимы, 608; 612
    хешированная база данных, 615
  разбивка конвертов, 679
  режим
    доставки, 678
    протоколирования, 685
    работы, 598; 627
  ретрансляция, 655; 656
  спам
    черные списки, 655; 661
  специальные пользователи, 671
  списки рассылки, 611; 616
  средство
    accept_unqualified_senders, 662
    accept_unresolvable_domains, 662
    access_db, 643; 658
    allmasquerade, 648
    always_add_domain, 641
    blacklist_recipients, 661
    delay_checks, 660
    dnsbl, 661
    genericstable, 644
    ldap_routing, 646
    limited_masquerade, 648
    local_lmtp, 651; 673
    local_procmail, 651
    loose_relay_check, 657
    mailertable, 644
    masquerade_entire_domain, 648
    masquerade_envelope, 648
    nocanonify, 642
    nouucp, 658
    nullclient, 650

promiscuous_relay, 657
redirect, 614; 641
relay_based_on_MX, 657
relay_entire_domain, 657
relay_hosts_only, 657
relay_local_from, 657
smrsh, 651; 673
use_cw_file, 640
virtusertable, 645
статистика работы, 683
таблицы, 642
файл переключения служб, 627
фальсификация почты, 676
фильтрация почты, 664
Set Group ID (бит смены идентификатора групп), 86
Set User ID (бит смены идентификатора пользователя), 86
SIMM (Single In-line Memory Module), 873
SuSE, 46
инсталляция, 282
Syslog, 260
безопасность, 758

T

TCP (Transmission Control Protocol), 325
TCP/IP, 325
пакет
поле TTL, 717
TIA (Telecommunications Industry Association), 412
TSM (Tivoli Storage Manager), 246

U

UDP (User Datagram Protocol), 325; 330
UPS (Uninterruptible Power Supply), 877
URL, 794

V

Veritas, 247
Virtual File System (VFS), 167

W

Web-сервер, 794
TUX, 802
распределение нагрузки, 797
Web-хостинг, 794

X

XORP (eXtensible Open Router Platform), 400
X Window System, 815

А

Автозагрузчик, 216
Авторское право, 1036
Адрес
IP, 336
Адресация
в стандарте IPv6, 346
пакетов, 334
Анализатор пакетов, 725
Аппаратные средства
обслуживание, 870
профилактическое, 873
условия эксплуатации, 874
Архивирование, 218
Атрибуты файлов, 125

Б

База данных
termcap, 939
terminfo, 939
Безопасность, 371; 1027
беспроводных сетей, 417
Библиотека
графических интерфейсных элементов, 816
интерфейсов информационных технологий, 987
ленточная, 216
Бит
setgid, 126
setuid, 87; 126
Sticky, 126
Блок, 952
Брандмауэр, 373; 776; 779
безопасность, 779
фильтрация пакетов, 776
на уровне служб, 776; 778
с учетом состояния, 778; 782

В

Виртуальная частная сеть, 783; 784
Виртуальный интерфейс, 802
конфигурирование, 803
Витая пара
неэкранированная, 411
Вычислительная сеть, 408
глобальная, 408
DSL, 422
Frame Relay, 421
ISDN, 422
локальная, 408
ATM, 420
Ethernet, 408
FDDI, 418
беспроводная, 416
виртуальная, 415
общегородская, 408

Г

Группа
  сетевая, 579

Д

Демон, 967
  amd, 977
  anacron, 203
  atd, 969
  automount, 557; 977
    главный файл, 558
    таблица назначений, 558; 559
  cron, 197; 207; 969
  crond, 969
  cupsd, 976
  dhcpd, 368; 982
  fcron, 203
  ftpd, 983
  gated, 399; 984
  getty, 936
  hald, 982
  httpd, 801; 985
  identd, 676
  imapd, 597; 608; 980
  in.fingerd, 984
  in.rlogind, 981
  in.rshd, 981
  in.telnetd, 981
  in.tftpd, 982
  inetd, 968; 969; 972
  init, 73; 968
    уровни выполнения, 968
  klogd, 976
  lockd, 545
  lpd, 977
  lwresd, 979
  mountd, 548; 973
  named, 471; 984
    безопасность, 512
    журнальная регистрация, 492; 525
    конфигурирование, 480
    начальный каталог, 483
    номер версии, 470
    отладка, 525
    уровни отладки, 530
    флаг -d, 530
    флаг -t, 512
    флаг -u, 512
  nfsd, 548; 551; 554
  nmbd, 978
  nscd, 565; 979
  ntpd, 985
  pop3d, 608
  popd, 980
  portmap, 548; 975
  pppd, 384
  routed, 399; 984
  rpc.bootparamd, 982
  rpc.lockd, 978
  rpc.mountd, 977
  rpc.nfsd, 977
  rpc.rquotad, 978
  rpc.statd, 978
  rpc.ypxfrd, 979
  rpciod, 978
  rquotad, 545
  rsyncd, 983
  sendmail, 979
  slapd, 621
  smbd, 978
  smtpd, 604; 980
  smtpfwdd, 604
  snmpd, 732; 983
  sshd, 771; 980
    регистрация без пароля, 456
  statd, 545
  syslogd, 262; 984
  talkd, 983
  udevd, 982
  xinetd, 969; 970
  ypbind, 575; 979
  yppasswdd, 580
  ypserv, 575; 979
  ypxfrd, 574
  обслуживания файлов, 977
  печати, 976
  электронной почты, 979
  ядра, 975
Диск
  жесткий, 216
  подключение, 169
  разбивка на разделы, 170
  форматирование, 169
Дисковая квота, 545
Диспетчер
  TSM, 246
  дисплеев, 816
  логических томов (LVM), 187
Дистрибутив, 44
Документы RFC, 329
Домен
  широковещательный, 410
Драйвер принтера, 840
драйвер терминала, 939
Драйвер устройства, 951
Дыры, 218

Ж

Жесткий диск, 216
Журнальный файл, 251

З

Загрузка
  автоматическая, 62
  в однопользовательском режиме, 71
  на персональном компьютере, 65
  начальная, 61
  ручная, 62
Загрузчик
  GRUB, 67
  LILO, 68
Зомби, 104

И

Идентификатор
  группы (GID), 97; 142
    по умолчанию, 142
    эффективный (EGID), 97
  пользователя (UID), 86; 97; 141
    эффективный (EUID), 97
  процесса (PID), 96
  родительского процесса (PPID), 96
Имя файла, 115
Инициализация ядра, 63
Инкапсуляция, 331
Инсталляция программного обеспечения, 53
Интерфейс
  Fibre Channel, 158
  IDE, 157; 165
  PATA, 158; 159
  SATA, 158; 160
  SCSI, 157; 158; 161; 165
  USB, 158; 946
  обратной связи, 336
Источник питания, 877

К

Кабель
  длина кабеля, 933
  прокладка, 424
  рекомендуемые поставщики, 429
  стандарты, 425
Кадр *См.* Фрейм, 331
Канал
  именованный, 124
Каталог, 122
  /etc/sysconfig, 77
  домашний, 143
  стандартный, 119
Команда, 51
  apt-ftparchive, 297
  apt-get, 293; 296; 297
    конфигурирование, 294; 295
  apt-proxy, 296
  arp, 351
  checksendmail, 684
  chkconfig, 582
  chmod, 128
  chown, 129; 672
  config term, 403
  cpio, 226
  crack, 765
  dd, 227
  df, 194; 554
  dig, 467; 474; 535
    определение версии BIND, 469
  dnskeygen, 513
  dnssec-keygen, 513; 518
  dnssec-signzone, 519
  domainname, 582
  dpkg, 289
  dump, 218
  echo, 955
  enable, 851
  export, 842
  exportfs, 549
    опция -a, 551
  fdisk, 188
  fetchmail, 608
  find, 116
  free, 890
  fsck, 178
  fuser, 117
  getfacl, 133
  grep, 845
  halt, 82
  hdparm, 176
  host, 474
  ifconfig, 337; 354; 803
  ifdown, 384
  insmod, 963
  iostat, 892
  iptables, 375; 779
    опция -A, 780
    опция -F, 780
    опция -i, 780
    опция -j, 780
    опция -P, 780
    список опций, 781
  kill, 102
  killall, 103
  kprinter, 865
  label, 239
  limit, 897
  locate, 54; 842
  logger, 270
  lpadmin, 849; 850
  lpoptions, 842; 843
  lpr, 843
  lprm, 842
  ls, 121; 952
  lsmod, 962
  lvcreate, 191
  lvextend, 194
  mail, 595; 596

mail.local, 596; 651; 673
mailstat, 652
mailstats, 683
makedbm, 574
makemap, 642
man, 51
mdadm, 189; 192
messages, 240
mii-tool, 357
mount, 116; 173; 552; 558; 915
  флаги монтирования, 553
mreport, 688
named-checkconf, 481
named-checkzone, 481
named-xfer, 507
ncftp, 572
ndc
  инструкция dumpdb, 532
  инструкция notrace, 530
  инструкция reload, 532
  инструкция trace, 530
  список инструкций, 531
netstat, 348; 359; 390; 719
  опция -a, 721
  опция -i, 720
  опция -n, 723
  опция -r, 722
  опция -s, 723
newaliases, 615
nfsstat, 555
nice, 104
nmap, 762
nslookup, 474
nsupdate, 509
passwd, 149
pine, 595
ping, 715
poweroff, 82
procinfo, 890
procmail, 596; 651
ps, 105
pvcreate, 190
rdist, 567
  управляющий файл, 568
rdump, 220
reboot, 82
renice, 104
restore, 222; 241
rndc, 496
  список инструкций, 531
rndc-confgen, 496
route, 358; 399
rsync, 570
run, 239
sar, 894
scp, 771
setfacl, 133
setserial, 934
sh, 596
show config, 402
showmount, 552
show running, 402
shutdown, 81
smbcontrol, 920
smrsh, 596; 673
spop, 597; 639
ssh, 567; 771; 822
ssh-keygen, 771
status, 244
stty, 940
su, 89
sudo, 178
swapon, 890
tar, 226
tcpdump, 726
telinit, 82; 937; 969
telnet, 795
top, 107
traceroute, 717
tset, 941
tune2fs, 167
umask, 130
umount, 554
uptime, 886
useradd, 137; 153
userdel, 154
usermod, 154
vgcreate, 191
vgdisplay, 191; 193
vmstat, 885
wget, 572
which, 53
xauth list, 820
xeyes, 823
ypinit, 580
yppasswd, 580
yppush, 575
ypset, 573
ypxfr, 574; 580
Коммутатор, 414
  беспроводной, 418
Компания SUSE, 46
Компиляция ядра, 959
Конфигурирование
  X-сервера, 823
  аппаратных терминалов, 935
  параметров ядра Linux, 955; 958
  программного обеспечения, 935
  сетевых принтеров, 849
Конфиденциальность, 1038
Концентратор, 413
Копирование
  резервное, 205
Криптография, 769

Л

Ленточный массив, 216

М

Маркер конца файла (EOF), 227
Маркетинг, 1002
Маршрутизатор, 390; 416
  Cisco, 401
Маршрутизация, 348
  таблица
    проверка, 722
Маска подсети, 337
Массив
  ленточный, 216
Мастер Add Printer, 865
Модем, 942
  дуплексный, 944
Модуль памяти SIMM, 873
Монитор, 872

Н

Начальная загрузка, 61
Неэкранированная витая пара (НВП), 411

О

Организация
  APNIC, 442
  CAIDA, 460
  CENTR, 441
  DMTF, 728
  IANA, 461
  ICANN, 437; 441
  ISC, 468
  LI, 1053
  LPI, 1053; 1056
  POSIX, 1047
  SAGE, 1053
  SANS, 1053
  USENIX, 1053
Останов системы, 80

П

Пакет, 331
  ADSM, 246
  Bacula, 228
  OpenOffice, 905
  Samba, 907
  TSM, 246
  Veritas, 247
  адресация, 334
Пароль, 750
  выбор, 752
  зашифрованный, 140
  устаревание, 753
Перезагрузка системы, 80
Переменная окружения DISPLAY, 819
Печать
  в KDE, 864
  в UNIX, 976
  язык
    PostScript, 837
Планировщик ввода-вывода, 894
Подсеть, 337
Пользователь
  daemon, 93
  nobody, 93; 547
  root, 89; 754
    в NFS, 547
  удаление, 152
Порт, 335
Принтер
  GDI, 858
  выбор принтера, 857
  сетевой, 859
  с последовательным и параллельным интерфейсом, 859
Программа
  Apache, 798
    запуск, 801
    инсталляция, 799
    конфигурирование, 800
  Berkeley DB, 615; 625; 642
  Ethereal, 727
  Eudora, 595
  Exim, 596; 689; 690
    конфигурирование, 690
  ghostscript, 639
  HylaFAX, 639
  Kerberos, 769
  Kickstart, 279
    создание сервера, 281
    файл конфигурации, 279; 281
  listmanager, 617; 619
  ListProc, 616; 619
  LISTSERV Lite, 617; 619
  Mailman, 616; 618
  Majordomo, 616; 617
  minicom, 944
  ndbm, 642
  Nessus, 764
  OpenSSH, 771
  Outlook, 595
  PGP, 677; 770
  QuickPage, 639
  SmartList, 617; 619
  smbclient, 914
  SpamAssassin, 665
  Squid, 808
    инсталляция, 809
  SSH, 771
  tcpd, 571; 974
  telnet, 899
  YaST, 282

Производительность, 885
Прокси-сервер, 296; 808
Протокол
  ARP, 330; 350
  BGP, 394
  CIDR, 341
  DHCP, 365; 982
  DVMRP, 398
  EIGRP, 394; 398
  ESMTP, 596
  FTP, 777
    анонимный сервер, 810
  HTTP, 794; 795; 844
    версия 1.1, 803
  ICMP, 330
  ICP, 809
  IGRP, 394; 398
  IMAP, 597; 607
  IPP, 976
  IPSEC, 783
  IPv6
    поддержка в BIND, 446; 463
  IS-IS, 395; 398
  LDAP, 582; 585
    безопасность, 588
    пакет OpenLDAP, 620
    применение в sendmail, 620; 646
  LMTP, 651
  MOSPF, 398
  NTP, 985
  OSPF, 395; 397
  PIM, 398
  POP, 597; 607
  PPP, 375
  PXE, 278; 982
  RDP, 903
  RIP, 393; 396
  RIP-2, 397
  SASL, 670; 677
  SMB, 907
  SMTP, 594; 596
    команды, 685
  SNMP, 728; 729
    агент NET-SNMP, 732; 733
    база MIB, 730
    база MIB-II, 730
    база RMON, 731
    групповое имя, 731
    дистанционный мониторинг, 731
    идентификаторы объектов, 730
    операции, 731
    прерывание, 731
  SSH, 822; 901
  SSL, 670
  TCP
    использование в NFS, 544; 553
  TCP/IP, 325; 329
  TLS, 670; 678
  UDP, 325; 330
    использование в NFS, 544; 553
  VNC, 903
  X11, 901
  маршрутизации, 392
Процесс, 95
  идентификатор процесса (PID), 96
  состояния процесса, 103
  ядра Linux, 64
Псевдотерминал (PTY), 952

P

Раздел, 170
  корневой, 171
  подкачки, 171
Разъем
  DB-9, 928
  DIN-8, 927
  RJ-45, 929
  RS-232C, 924
Резервное копирование, 205
  безопасность, 760
Ретрансляция кадров, 421

C

Сборка ядра Linux, 957
Сегмент, 331
Сервер
  CUPS, 841
    администрирование, 847
  Samba
    отладка, 919
  X Server, 901
Сетевое конфигурирование
  в Debian, 365
  в Fedora, 362
  в Red Hat, 362
  в SUSE, 364
  в Ubuntu, 365
Сеть
  беспроводная, 333
  виртуальная частная, 373
  поиск неисправностей, 714
  управление, 713
Сигнал, 99
Символическая ссылка, 124
Система
  X Window System, 815
  запроса открытых документов, 1021
  регистрации событий Syslog, 260
Системное администрирование, 1002; 1004
Системный вызов
  chroot, 810
Слежка (snooping), 351
Служба
  DDS, 421

LDAP, 911
NIS, 911
каталогов, 583
Сокет
локальный, 123
Спам
черные списки, 661
Список контроля доступа (ACL), 132
Среда
GNOME, 833
KDE, 832
Стандарт Йоста, 930
Стек сетевых протоколов Linux, 361
Страница, 888
Суперблок, 172
Суперпользователь, 87; 754
в NFS, 547
Сценарий запуска, 72
Debian, 80
Fedora, 76
Red Hat, 76
SUSE, 78
Ubuntu, 80

Т

Таблица страниц, 888
Терминал
защищенный, 759
Типы файлов, 120
Топология Ethernet, 410
Троянский конь, 761

У

Утилита
GRUB, 66
LILO, 66
logrotate, 259
pr, 854
SEC, 273
smbpasswd, 909
sudo, 90
swatch, 272
wvdial, 380
Учетная запись
отключение, 153
управление, 153

Ф

Файл
.forward, 614
.rhosts, 759
/etc/exports, 549
/etc/fstab, 555
/etc/group, 146; 758
/etc/hosts, 353
/etc/hosts.equiv, 759
/etc/inetd.conf, 972
/etc/inittab, 937
/etc/mail/aliases, 609; 611
/etc/named.conf, 510
инструкция acl, 489
инструкция controls, 496
инструкция include, 482
инструкция key, 489
инструкция logging, 492; 526
инструкция options, 482
инструкция server, 490
инструкция trusted-keys, 490
инструкция view, 497
инструкция zone, 492; 495
пример, 499; 500
список соответствия адресов, 482
список управления доступом, 489; 511
/etc/netgroup, 579
/etc/nsswitch.conf, 539; 577; 627
/etc/passwd, 138; 573; 758
защита, 751; 754
/etc/resolv.conf, 478
/etc/rsync.conf, 571
/etc/shadow, 143; 752; 758
защита, 751
/etc/yp.conf, 574
/var/log/wtmp, 257
bacula-dir.conf, 232
bacula-sd.conf, 236
bconsole.conf, 237
crontab, 198
dhcpd.conf, 368
PPD, 845
services, 974
атрибуты файлов, 125
блокировка, 545
журнальный, 251
имя, 115
копирование, 566
обычный, 122
переключения служб, 577; 627
права доступа, 757
бит SUID, 757
принудительная рассылка, 566; 567; 570
путь к файлу, 115
рассылка по запросу, 566; 572
сетевой конфигурационный, 352
совместное использование, 564
типы файлов, 120
устройства, 952
Файловая система, 115
autofs, 557; 558
ext2fs, 167
ext3fs, 167
fsck, 179
JFS, 168
MS DES, 913
NFS, 93; 201

ReiserFS, 168
smbfs, 914
sysfs, 953
XFS, 168
архивирование, 218; 225
виртуальная, 167
восстановление, 178; 224; 225
монтирование, 116; 173
автоматическое, 557
организация, 118
проверка, 178
размонтирование, 116
сетевая, 543; 546
демонтирование, 554
монтирование, 552; 554
чистка, 201
экспорт, 548; 549
опции, 550
Фильтр, 846
Формат PDF, 838
Форматирование диска, 169
Фрейм, 331
Функция
closelog, 271
gethostbyname(), 473
openlog, 270

Ш

Шифрование, 1036

Э

Электронная почта, 603
агент
доставки, 594; 596; 638
доступа, 594; 597
подачи почты, 594; 597
пользовательский, 594; 595
транспортный, 594; 596
адресация, 599
почтовые каталоги, 606
почтовые псевдонимы, 608
сервер, 604
сообщение
заголовки, 599
маршрутизация, 456
структура, 598
хранилище, 596
стандарт MIME, 595

Я

Ядро, 949
адаптация ядра, 950
инициализация, 63
компиляция ядра, 959
конфигурирование
на почтовом сервере, 682
процессы ядра, 63
сборка ядра Linux, 957
Язык принтера
PCL, 838
PDF, 838
PJL, 839
PostScript, 837
XHTML-Print, 839
Ячейка, 420

Научно-популярное издание

Эви Немет, Гарт Снайдер, Трент Хейн

Руководство администратора Linux
2-е издание

Верстка *Т. Н. Артеменко*
Художественный редактор *С. А. Чернокозинский*

ООО “И. Д. Вильямс”
127055, г. Москва, ул. Лесная, д. 43, стр. 1

Подписано в печать 07.12.2010. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 86,43. Уч.-изд. л. 76,30.
Доп. тираж 1000 экз. Заказ № 0000.

Отпечатано по технологии CtP
в ОАО “Печатный двор” им. А. М. Горького
197110, Санкт-Петербург, Чкаловский пр., 15.

Book_LinuxAdmHand.indb 1072 17.04.2007 13:56:55