По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru-Книги России» единственный легальный способ получения данного файла с книгой «FreeBSD. Подробное руководство» (ISBN 5-93286-066-9) – покупка в Интернет-магазине «Books.Ru-Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (www.symbol.ru), где именно Вы получили данный файл.

Absolute BSD

The Ultimate Guide to FreeBSD

Michael Lucas

NO STARCH
PRESS

HIGH TECH

FreeBSD

Подробное руководство

Майкл Лукас

Санкт-Петербург — Москва
2004

Серия «High tech»

Майкл Лукас

FreeBSD. Подробное руководство

Перевод Ю. Кунивера

Главный редактор	*А. Галунов*
Зав. редакцией	*Н. Макарова*
Научный редактор	*Ф. Торчинский*
Редактор	*В. Овчинников*
Корректор	*С. Беляева*
Верстка	*Н. Гриценко*

Лукас М.
FreeBSD. Подробное руководство. – Пер. с англ. – СПб: Символ-Плюс, 2004. – 616 с., ил.
ISBN 5-93286-066-9

«FreeBSD. Подробное руководство» – всеобъемлющее руководство по FreeBSD – мощной, гибкой и бесплатной операционной системе семейства UNIX, выбранной многими предприятиями в качестве серверной платформы.

Прочитав книгу, вы сможете использовать FreeBSD для предоставления сетевых сервисов, научитесь управлять системами FreeBSD, поддерживать их и накладывать «заплатки». Руководство охватывает установку системы, работу в сети, вопросы безопасности, производительность системы, тонкую настройку ядра, файловые системы, SMP, проведение обновлений, устранение неполадок и управление программным обеспечением.

Руководство написано одним из активных участников проекта FreeBSD и адресовано администраторам UNIX, у которых назрела потребность в сборке и конфигурировании выделенных серверов FreeBSD. Книга будет интересна и пользователям, собирающимся запускать FreeBSD на своем рабочем компьютере или комбинировать настольные/серверные системы.

ISBN 5-93286-066-9
ISBN 1-886411-74-3 (англ)

© Издательство Символ-Плюс, 2004
Authorized translation of the English edition © 2002 No Starch Press, Inc. This translation is published and sold by permission of No Starch Press, Inc., the owner of all rights to publish and sell the same.

Все права на данное издание защищены Законодательством РФ, включая право на полное или частичное воспроизведение в любой форме. Все товарные знаки или зарегистрированные товарные знаки, упоминаемые в настоящем издании, являются собственностью соответствующих фирм.

Издательство «Символ-Плюс». 199034, Санкт-Петербург, 16 линия, 7,
тел. (812) 324-5353, edit@symbol.ru. Лицензия ЛП N 000054 от 25.12.98.
Налоговая льгота – общероссийский классификатор продукции
ОК 005-93, том 2; 953000 – книги и брошюры.
Подписано в печать 01.04.2004. Формат 70x100 1/16 . Печать офсетная.
Объем 38,5 печ. л. Тираж 3000 экз. Заказ N
Отпечатано с диапозитивов в Академической типографии «Наука» РАН
199034, Санкт-Петербург, 9 линия, 12.

Посвящается, как всегда, Лиз

Благодарности

Хочу выразить признательность всем членам сообщества FreeBSD за их нелегкий труд, преданность и дружелюбие. Система FreeBSD спасала меня много раз, и благодаря ей я узнал, как на самом деле работают компьютеры и Интернет. Однако мне еще есть о чем поговорить с президентом компании, разрабатывающей программное обеспечение, поскольку я потратил много часов, обсуждая FreeBSD с руководителями проектов.

В сообществе есть несколько человек, заслуживших особую признательность за критику книги, которую вы держите в руках. Вот они – Сильвестр Адам (Szilvester Adam), Джон Болдуин (John Baldwin), Уилко Булт (Wilko Bulte), Крис Диллон (Chris Dillon), Джорджос Керамидас (Giorgos Keramidas), Крис Найт (Chris Knight) и Джоэл Уилсон (Joel Wilsson). Все ошибки в этой книге допущены мной несмотря на все их усилия.

Сотрудники No Starch Press также заслуживают искренней признательности за выход этой книги в свет. Им пришлось значительно переработать рукопись, чтобы текст прилично выглядел на печатных страницах. Спасибо, постараюсь в следующий раз облегчить вашу работу. Также я хочу поблагодарить Криса Коулмана (Chris Coleman), моего редактора на *OnLamp.com*, который, прежде всего, примирил меня и No Starch Press.

Больше всего я признателен моей жене Лиз за ее терпение и поддержку, пока я месяцами сидел в углу и невнятно бормотал, работая над этой книгой.

Майкл Лукас (Michael Lucas),
Сент-Клер Шорз, Мичиган

Оглавление

Предисловие 13
Введение 14

1. Установка 30
Аппаратное обеспечение FreeBSD 30
Загрузка FreeBSD 31
Настройка аппаратных средств 34
Непосредственная установка FreeBSD 35
Разбиение диска 39
root 41
Пространство для свопинга 41
/var, /usr/ и /home 43
Второй жесткий диск 43
Настройка после инсталляции 47
Перезапуск 55
Замечание о редакторах 56

2. Как получить помощь 57
Почему не почтой с самого начала? 57
Позиция FreeBSD 58
Страницы руководства 59
Документация FreeBSD 62
Архивы почтовых рассылок 62
Другие веб-сайты 63
Использование ресурсов FreeBSD, связанных с принятием решений 64

3. Прочтите это раньше, чем что-нибудь испортите! (Резервное копирование и восстановление) 70
Резервное копирование системы 71
Накопители на лентах 71
Управление накопителем на ленте 72
Программы для создания резервных копий 75
Восстановление из архива 82

Запись происходящих событий . . . 85
Управление версиями . . . 86
Однопользовательский режим . . . 92
Диск восстановления . . . 94

4. Эксперименты с ядром . . . 96

Что такое ядро? . . . 96
Конфигурирование ядра . . . 97
Загрузка и выгрузка модулей в многопользовательском режиме . . . 105
Настройка ядра . . . 105
Создание ядра . . . 115
Загрузка с запасного ядра . . . 118
Добавления в ядро . . . 119
Настройка производительности ядра . . . 121
Распространение ядра . . . 123

5. Работа в сети . . . 124

Сетевые уровни . . . 124
Сеть на практике . . . 128
Ethernet . . . 131
Подсоединение к сети Ethernet . . . 139

6. Обновление FreeBSD . . . 147

Версии FreeBSD . . . 147
Методы обновления . . . 152
Сборка локального сервера CVSup . . . 166

7. Организация защиты системы . . . 173

Кто враг? . . . 174
Сообщения, относящиеся к безопасности FreeBSD . . . 175
Установка профилей безопасности . . . 177
root, группы и права доступа . . . 178
Изменение прав доступа . . . 182
Флаги файлов . . . 186
Уровни безопасности . . . 189
Программы, которые можно взломать . . . 192
Собрать все вместе . . . 196

8. Расширенные средства защиты . . . 197

Управление трафиком . . . 197
TCP Wrappers . . . 198
Фильтрация пакетов . . . 207

Клетка . . . 219
Мониторинг системы защиты . . . 227
Если система взломана . . . 228

9. Каталог /etc . . . 229

Разнообразие файлов /etc . . . 229
Файлы по умолчанию . . . 230

10. Делаем систему полезной . . . 276

Сборка программного обеспечения . . . 277
Система «портов» и «пакетов» . . . 279
Применение пакетов . . . 284
Применение «портов» . . . 293
Обновление «портов» и пакетов . . . 302

11. Расширенное управление программным обеспечением . . . 307

Сценарии запуска и останова . . . 308
Управление совместно используемыми библиотеками . . . 310
Запуск программного обеспечения из чужой ОС . . . 314
Установка и включение режима Linux . . . 319
Использование нескольких процессоров – SMP . . . 323

12. Поиск хостов с помощью DNS . . . 329

Как работает DNS . . . 329
Основные инструменты DNS . . . 330
Настройка клиента DNS: распознаватель . . . 336
Источники информации DNS . . . 337
Пример реально существующей зоны . . . 350
Активизация изменений . . . 353
Запуск named во время начальной загрузки . . . 354
Проверка DNS . . . 354
Защита named . . . 354
Управление порядком предоставления информации . . . 356
Дополнительная информация о BIND . . . 356

13. Управление малыми сетевыми сервисами . . . 357

Управление пропускной способностью . . . 357
Шифрование открытым ключом . . . 362
SSH . . . 369
Системное время . . . 376
Демон inetd . . . 379

14. Почтовые службы . . . 383

Обзор электронной почты . . . 383
Почтовые программы . . . 387
POP3 . . . 396

15. Веб- и FTP-сервисы . . . 404

Как работает веб-сервер . . . 404
Веб-сервер Apache . . . 405
Виртуальный хостинг . . . 425
.NET во FreeBSD . . . 429
FTP . . . 430

16. Файловые системы и диски . . . 437

Файлы устройств . . . 437
Файл /etc/fstab . . . 439
Основы организации диска . . . 441
Fast File System . . . 442
Что монтировано сейчас? . . . 446
«Грязные» диски . . . 446
Монтирование и демонтирование дисков . . . 448
Soft Updates . . . 449
Кэширование каталогов в виртуальной памяти . . . 451
Монтирование неродных файловых систем . . . 451
Права доступа в файловых системах . . . 453
Съемные носители и /etc/fstab . . . 454
Создание дискеты . . . 454
Основы SCSI . . . 456
FreeBSD и SCSI . . . 458
Добавление новых жестких дисков . . . 460

17. RAID . . . 465

Сравнение программных и аппаратных RAID . . . 465
Уровни RAID . . . 466
Программные RAID . . . 466

18. Производительность системы . . . 482

Ресурсы компьютера . . . 482
Дисковый ввод/вывод . . . 484
Сетевая пропускная способность . . . 484
Процессор и память . . . 484
Когда своп некстати . . . 493
Настройка производительности на практике . . . 497
Извлеченные уроки . . . 503

19. Как обстоят дела? 504

Письма о состоянии 504
Протоколирование с помощью syslogd 505
Отчеты с помощью SNMP 515
Долгосрочный мониторинг с помощью MRTG 525

20. Аварии и паника в системе 536

Что вызывает панику? 536
Ответные действия при панике 538
Применение dump 543
Последовательные консоли 551
Предоставление отчета о проблеме 557

21. Настольная система FreeBSD 563

Доступ к разделяемым файловым ресурсам 563
Доступ клиентов Windows к разделяемым файлам UNIX 570
Доступ к серверам печати 570
Локальные принтеры 572
X: графический интерфейс 573
Настольные приложения 575

Послесловие 581

Приложение. Полезные sysctl MIB 586

Алфавитный указатель 595

Предисловие

Двадцать пять лет. Боже мой, неужели прошло столько времени? Первый «выпуск» BSD появился на свет в 1976 благодаря усилиям группы по исследованию компьютерных систем (CSRG) Калифорнийского университета в Беркли. Этот и последующие «выпуски» BSD оказали значительное влияние на все последующие системы UNIX, включая Linux и System V компании AT&T. В них были реализованы новшества и добавлены все «отсутствующие части», которых не было в UNIX. Из BSD берут свое начало такие возможности, как виртуальная память, сети TCP/IP, управление заданиями и даже почтенный экранный редактор vi (до этого в системах был только ed(1)). Системе BSD обязаны своим существованием не только операционные системы, но и стандарты POSIX и X/Open – действительно важный «маленький проект»!

В 1992 году мы начали работу над FreeBSD – проектом, о котором много говорится в этой книге и который способствовал продолжению этой работы после роспуска группы CSRG. Казалось, что проект BSD, несмотря на всю свою историю и перспективы, подходит к концу. С таким положением не могли смириться многие фанаты BSD, и я рад сообщить, что в этой ситуации они объединили свои усилия.

До заката BSD далеко. Последние 10 лет наблюдается лавинообразный рост сообщества BSD. Операционную систему FreeBSD применяют самые значимые компании и сайты в Интернете. Они расширяют просторы Сети и демонстрируют высокое время наработки на отказ, делая аббревиатуру BSD синонимом системы с высокой производительностью, безопасностью и надежностью в индустрии сервисов Интернета.

Кроме того, успех FreeBSD не сводится только к серверам. Компания Apple приняла FreeBSD в качестве ключевой открытой технологии для разработки своей операционной системы. В результате появилось новое поколение энтузиастов. Многие из них никогда не считали себя пользователями UNIX, однако сейчас они наслаждаются преимуществами мощной операционной системы в сочетании с легендарным пользовательским интерфейсом Apple и первоклассным набором приложений. Даже пресытившиеся эксперты UNIX были поражены тем, во что выросла система BSD. Подозреваю, что сегодня она превзошла самые фантастические мечты своих создателей.

Неважно, являетесь ли вы экспертом UNIX или никогда не сталкивались с этой системой ранее. Эта книга послужит для вас отличным введением в уникальный и впечатляющий мир BSD. Если вы получите от системы хотя бы половину того удовольствия, какое получил я, то вас многое ждет впереди!

Джордан Хаббард (Jordan Hubbard),
соучредитель проекта FreeBSD

Введение

Добро пожаловать во «FreeBSD. Подробное руководство»! Эта книга представляет собой всеобъемлющее руководство для администраторов UNIX, у которых назрела необходимость в сборке и конфигурировании выделенных серверов FreeBSD. Книга будет интересна и пользователям, собирающимся запускать FreeBSD на своем рабочем компьютере или комбинировать настольные/серверные системы.

Прочитав книгу, вы сможете использовать FreeBSD для предоставления сетевых сервисов. Вы научитесь управлять системами FreeBSD, поддерживать их и накладывать «заплатки». Вы получите базовое представление о работе в сети, безопасности системы и управлении программным обеспечением. В книге обсуждается версия 4. На момент выхода данной книги эта версия FreeBSD рекомендована для широкого использования. Большая часть книги также применима для предшествующих и последующих версий. Многие материалы этой книги также применимы к NetBSD и OpenBSD.

Что такое FreeBSD?

FreeBSD – это операционная система, подобная UNIX,[1] которая свободно доступна в Интернете. Она широко применяется в компаниях-провайдерах услуг Интернета (ISP, Internet service provider), во встроенных устройствах и в любом другом месте, где важна надежность. Ее корни уходят непосредственно в первоначальную систему UNIX, выпущенную AT&T в 1970-х.

Много лет назад компании AT&T потребовалось программное обеспечение для организации своего бизнеса. Однако ей не разрешалось конкурировать с компаниями компьютерной индустрии. В результате AT&T лицензировала различные куски программного обеспечения и исходный код, предоставив его университетам по очень низкой цене. Студенты университетов получили доступ к отличной технологии. Они могли читать исходный код и изучать его работу. Взамен AT&T получила бесплатную площадку для экспериментов и поколение специалистов по вычислительной технике, выросших на оборудовании AT&T. Все были довольны. Система UNIX была самым известным программным обеспечением, распространяемым по этому плану лицензирования.

---

[1] Почему подобная UNIX? Что ж, слово UNIX является торговой маркой организации The Open Group. Чтобы та или иная операционная система была сертифицирована как «UNIX», надо заплатить большие деньги этой организации. Поскольку FreeBSD разрабатывается на некоммерческой основе, это маловероятно.

Как возникла FreeBSD?

Исходная система UNIX проигрывает в сравнении с современными операционными системами. Однако поскольку так много студентов получили исходный код UNIX и стольким преподавателям потребовались проекты для своих студентов, то благодаря совместным усилиям система UNIX была вскоре улучшена. Мало-помалу создавались полезные команды. Была добавлена возможность управлять запуском программ (управление заданиями, *job control*). Появилась файловая система, поддерживавшая функции, которые мы воспринимаем сейчас как данность. За многие годы были извлечены и заменены целые блоки оригинальной операционной системы UNIX.

Различные университеты, работавшие над UNIX, делились друг с другом своими улучшениями и расширениями. Группа по исследованию компьютерных систем (Computer Systems Research Group, CSRG) Калифорнийского университета стала центральным хранилищем улучшений кода UNIX. Этот код мог получить каждый обладатель действительной лицензии AT&T UNIX.

Итоговая коллекция «заплаток» для UNIX получила название Berkeley Software Distribution, или BSD UNIX. (Статус Беркли не пострадал от того, что Управление перспективных исследовательских программ (Defense Advanced Research Projects Agency, DARPA) профинансировало CSRG для реализации TCP/IP в UNIX.)

Разработка была долгой. Посмотрев на информацию об авторских правах FreeBSD, можно увидеть следующее:

```
Copyright 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
```

Да, 15 лет работы – это в разработке программного обеспечения целая жизнь. По существу, в оригинальную систему UNIX было внесено столько изменений, что группа CSRG обнаружила почти полную замену созданного ее сотрудниками и помощниками кода UNIX. От продукта AT&T осталось совсем мало.

Лицензия BSD: BSD становится общедоступной

В конце концов финансирование CSRG прекратилось. В 1992 году после некоторых споров в Калифорнийском университете код был открыт широкой публике. Такая лицензия получила название лицензии BSD. Сегодня в ней есть три статьи, которые можно суммировать так:

- Не заявляйте о том, что вы написали этот код.
- Не обвиняйте нас за ошибки в коде.
- Не используйте наше имя для продвижения своего продукта.

(Первоначальная лицензия требовала упоминать при каждом использовании программного обеспечения, что авторскими правами на него обладает Калифорнийский университет. Через несколько лет это требование было от-

менено. Сегодня все желающие могут использовать код BSD, не объявляя об этом и никого не уведомляя.)

Возможно, лицензия BSD – самая либеральная из всех когда-либо существовавших. BSD можно свободно включать в патентованные продукты, бесплатные продукты и продукты с открытым исходным кодом. Лицензию можно распечатать на перфокартах и покрыть ими лужайку.

Иногда при обсуждении лицензии BSD упоминается не «copyright» (авторское право), а «copycenter» (копировальный центр) – «возьмите эту лицензию в копировальный центр и напечатайте для себя несколько копий». Неудивительно, что некоторые компании, например Sun Microsystems, так за нее ухватились – она же бесплатна.

Рождение современной FreeBSD

Однако в эпоху расцвета CSRG работа над UNIX быстро перешла к AT&T. AT&T приняла участие в распространении BSD UNIX и интегрировала эту систему со своими системами UNIX, а затем внесла в нее изменения и выпустила результат под новой лицензией.

Это отлично подходило для AT&T до великого распада, когда праматери всех телефонных компаний неожиданно разрешили конкурировать на рынке программного обеспечения. У AT&T была одна значительная собственность: высококлассная операционная система, которая была отлажена тысячами специалистов со всего мира. AT&T стала успешно продавать UNIX предприятиям за большие деньги, поддерживая при этом отношения с университетами, которые предоставили ей такую усовершенствованную операционную систему.

«Выпуск» BSD, созданный в Беркли в 1992 году, был с большим неудовольствием встречен дочерней компанией AT&T – USL (UNIX System Laboratories). Почти сразу же представители этой компании вызвали нескольких пользователей и университет в суд. Компания USL заявила, что разработчики в Беркли завладели их интеллектуальной собственностью. Калифорнийский университет ответил, что это его интеллектуальная собственность. Тем временем многие компании добыли код, выпущенный Беркли, и стали создавать на его основе коммерческие и бесплатные продукты. Одним из таких продуктов была система 386BSD, которая со временем стала ядром FreeBSD 1.0.

В 1994 году после двухлетних разбирательств суд пришел к мнению, что значительная часть кода в AT&T UNIX действительно могла быть взята почти наверняка только из BSD. Источником разногласий были только полдюжины файлов. Чтобы положить конец этим спорам, часть этих файлов была подарена, а остальные – запатентованы. К сожалению, система FreeBSD 1.X содержала некоторые из этих файлов, поэтому пользователи BSD неистово работали, чтобы воссоздать отсутствующие компоненты.

После того как все утихло, была выпущена новая версия UNIX под названием BSD 4.4-Lite. Последующее обновление, система BSD 4.4-Lite2, является

бабушкой текущего кода FreeBSD, а также прародительницей многих других операционных систем, таких как NetBSD, OpenBSD и Mac OS X.

Сегодня FreeBSD применяется по всему миру наиболее значительными и заметными компаниями, деятельность которых связана с Интернетом. Так, на момент написания этих строк Yahoo! почти полностью работает на системах FreeBSD. «Малышка Bell» US West применяет FreeBSD для поддержки операций в Интернете. IBM, Nokia и многие другие производители аппаратных средств применяют FreeBSD во встроенных системах, где об этом даже трудно догадаться.

По существу, если компании необходима серьезная полоса пропускания в Интернете, она наверняка запускает FreeBSD. Машины FreeBSD окружают вас; возможно, вы не видите их потому, что аварии на них случаются редко.

Разработка FreeBSD

Существует старая поговорка, согласно которой управление программистами подобно выгулу группы котов. Однако что бы вы ни думали, разработчики FreeBSD отлично работают вместе в составе команды FreeBSD. И, в отличие от некоторых других проектов, вся разработка FreeBSD ведется открыто. Систему FreeBSD разрабатывают две группы: *создатели* (*committers*) и *помощники* (*contributors*).

Создатели

Сегодня в состав команды FreeBSD входит почти 300 разработчиков, или *создателей*. Они имеют доступ на чтение/запись к основному репозитарию исходного кода FreeBSD и могут разрабатывать, отлаживать и улучшать любую часть кода по своему усмотрению.

Чтобы присоединиться к плеяде разработчиков FreeBSD, подпишитесь на почтовую рассылку *FreeBSD-hackers@FreeBSD.org*, содержащую большую часть обсуждений технических вопросов. Технические вопросы разбиты на более специфические почтовые рассылки – например, разработка сетевых служб обсуждается в *FreeBSD-net@FreeBSD.org*. Команда FreeBSD обсуждает технические вопросы также по нескольким каналам IRC. Посетители и «слушатели» приветствуются, если они никому не мешают. (Да, интернет-чат можно применять и в полезных технических целях!) Создатели отвечают за поддержку работоспособного состояния FreeBSD, добавление новых функций и оценку заплаток от помощников. Большинство таких разработчиков – добровольцы; лишь немногие получают деньги за свою усердную работу.

Помощники

В дополнение к команде создателей сообщество FreeBSD включает в себя тысячи помощников. Помощники не связаны с репозитарием операционной системы; они просто представляют заплатки на рассмотрение создателей. Создатели оценивают представленный код и решают, принять его или от-

клонить. Помощник, который постоянно представляет приемлемый код, зачастую получает от создателей предложение самому стать создателем.

Например, несколько лет я был помощником. Всякий раз, когда я чувствую, что зря прожил жизнь, я захожу на веб-страницу FreeBSD и смотрю, какая часть моей работы была принята создателями и применяется тысячами пользователей. (Это помогает. В какой-то степени.) Однако между написанием этой книги и получением отзыва от редактора у меня было свободное время. Я потратил его на подготовку заплаток для FreeBSD FAQ. В конце концов участники Проекта FreeBSD обратились ко мне с предложением стать создателем. Сначала я отказался, однако в конечном итоге позволил нескольким разработчикам убедить меня.[1]

Пользователи

Наконец, FreeBSD имеет множество пользователей, однако достоверно оценить их количество невозможно. В конце концов, вы можете бесплатно скачать всю систему FreeBSD и при этом никогда не регистрироваться, не обновлять систему и не подписываться на почтовую рассылку.

По приблизительным оценкам от 5 до 10 процентов машин в Интернете запускают BSD. Это 5–10 процентов всех систем, подключенных к Интернету, включая бесчисленные системы Windows, расположенные на офисных столах. Если не считать эти системы и принимать в рассмотрение только серверы Интернета, то процент возрастет.

Поскольку FreeBSD, безусловно, наиболее популярная система BSD с открытым кодом, это количество выглядит значительным. А поскольку один сервер FreeBSD может обслуживать сотни и тысячи доменов Интернета, то количество сайтов, задействующих FreeBSD, несоизмеримо с количеством серверов.

Другие системы BSD

FreeBSD – наиболее популярная система BSD, но не единственная. Система BSD 4.4-Lite породила различные проекты, у каждого из которых есть свое назначение.

NetBSD

Система NetBSD во многом подобна FreeBSD, а их команды делят между собой разработчиков и сам код. Основное назначение проекта NetBSD – предоставить операционную систему, которую можно перенести на любую аппаратную платформу.

Таким образом, NetBSD работает на VAX, устройствах PocketPC и высококлассных серверах Alpha, а также в Compaq iPaq. Она работает даже на аппаратных средствах, которые еще не существуют, – на момент написания

[1] Возможно, когда-нибудь я прощу Уилла, Уилко и Брюса. Однако я *никогда* не позволю им повернуть все вспять.

этих строк AMD Sledgehammer полностью поддерживался, хотя достать образец чипа было невозможно. Теперь и *это* переносимо.

Код NetBSD специально лицензирован для свободного повторного использования – подобно первоначальному исходному коду BSD 4.4-Lite, на котором она основана.

OpenBSD

OpenBSD ответвилась от NetBSD в 1996 году с целью стать самой безопасной BSD. OpenBSD стала первой системой, в которой была реализована поддержка криптоаппаратуры (она позволяет шифровать и расшифровывать информацию с замечательной скоростью). Разработчики OpenBSD гордятся тем, что их первоначальная система удаленно не взламывалась свыше четырех лет.

Команда OpenBSD тщательно изучила весь базовый код BSD, залатав большинство потенциальных «дыр» в защите (если не все), прежде чем они проявились. Однако OpenBSD не настолько проста в использовании и дружелюбна, как FreeBSD.

BSD/OS

BSD/OS, выпускаемая Wind River Systems, – это коммерческая операционная система с закрытым исходным кодом, очень похожая на FreeBSD. Некоторые производители аппаратных средств не выпускают спецификации на аппаратные средства без соглашения о нераскрытии информации, поэтому разработчики бесплатных операционных систем не могут разработать драйверы для таких устройств. BSD/OS поддерживает большую часть таких аппаратных средств.

Значительная часть кода BSD/OS доступна создателям FreeBSD, а FreeBSD впитывает улучшения BSD/OS, которые не нарушают соглашения о неразглашении.

Mac OS X

Mac OS X? Все верно. Большие фрагменты FreeBSD были включены в Mac OS X компании Apple. Если вы ищете стабильную операционную систему с дружелюбным интерфейсом и мощной основой, система Mac OS X вам, бесспорно, подойдет. Хотя FreeBSD предоставляет замечательную настольную среду для профессионалов-компьютерщиков, я не установил бы ее для своей бабушки. Не раздумывая, я установил бы ей Mac OS X и был бы уверен, что поступаю правильно.

Впрочем, Mac OS X включает в себя функции, которые вовсе не обязательны для сервера Интернета. Эта система работает только на платформе Apple, поэтому я не рекомендую ее для недорогих, высокомощных серверов.

Исходного кода пользовательского интерфейса Mac OS X нет в свободном доступе, но можно изучить базовый код BSD, использованный для этой операционной системы, и ядро Mach; компания Apple выпустила их под кодовым названием Darwin.

Другие UNIX

Существует несколько других операционных систем UNIX. Некоторые из них даже «арендуют» торговую марку UNIX, поэтому они могут именовать себя таким образом. Безусловно, список таких систем значителен, однако мы коснемся лишь основных.

Solaris

Наиболее известная система UNIX – это Solaris компании Sun Microsystems. Solaris работает на высококлассной аппаратной платформе, которая поддерживает десятки процессоров и кучу (gobs) дисков. (Да, «gobs» – это технический термин.) Solaris применяется многими приложениями уровня предприятия, такими как Oracle.

В основном Solaris запускается на аппаратной платформе SPARC, которая выпускается компанией Sun. Поскольку Sun контролирует как аппаратное, так и программное обеспечение, ее системы поддерживают много интересных функций, например возможность «горячей» замены модулей памяти и материнских плат.

AIX

Другой конкурент UNIX – система AIX компании IBM. AIX знаменита журналируемой файловой системой, которая регистрирует все дисковые транзакции по мере их выполнения. Она позволяет без особого труда восстановить систему после аварий, обеспечивая высокую надежность. AIX вобрала в себя значительную часть кода BSD.

Linux

Linux – это клон UNIX, написанный с нуля в последние десять лет или около того. Система Linux во многом подобна BSD, хотя BSD обладает намного большим наследием и лучше подходит для коммерческого использования, чем Linux. Linux включает в себя требование, гласящее, что все изменения, выполненные для коммерческих продуктов, должны вноситься обратно в Linux. BSD не имеет такого ограничения.

У многих пользователей UNIX возникает ощущение конфликта между лагерями BSD и Linux. Однако если копнуть немного глубже, можно обнаружить, что большинство разработчиков этих платформ открыто и дружелюбно взаимодействуют друг с другом. Лишь узкий круг пользователей и малая часть разработчиков становятся источником трений.

IRIX, HPUX и так далее

Есть и другие системы UNIX, например IRIX компании Silicon Graphics (солидная система UNIX для графических приложений) и HP-UX компании Hewlett-Packard, популярная на больших предприятиях. Многие высоко-

классные программные продукты, такие как Informix, специально разработаны для HP-UX.

Если оглядеться вокруг, можно также обнаружить меньших конкурентов, таких как SCO и UnixWare. Они вовсе не являются неважными, просто они не настолько популярны. Можно также встретить старые ископаемые, такие как A/UX компании Apple и Xenix компании Microsoft. (Да, Microsoft была лицензированным поставщиком UNIX, но это было очень давно.) В конце концов система Xenix была продана компании SCO и превратилась в SCO UNIX.

Сильные стороны FreeBSD

Как же после всего этого можно охарактеризовать FreeBSD?

Переносимость

Цель FreeBSD – предоставить свободно распространяемую операционную систему, способную работать на популярных аппаратных платформах. И хотя безопасность системы жизненно важна, основное назначение FreeBSD состоит в том, чтобы работать на тех из них, которые преобладают в различных компаниях. Сегодня это системы, совместимые с Intel x86 (386, 486, Pentium I–IV, Celeron и AMD). Кроме того, FreeBSD поддерживает процессор Alpha. Ведутся работы по поддержке новых микросхем IA64 компании Intel, 64-разрядных микросхем AMD, PowerPC компании Motorola, а также SPARC компании Sun. (Эти платформы новые; аппаратные средства уже появляются либо только сейчас становятся достаточно популярными, чтобы переносить на них систему.)

Мощь

Поскольку FreeBSD адекватно работает на аппаратной платформе 386, она достаточно хорошо работает на современных компьютерах. Довольно приятно работать с системой, не требующей процессора Pentium III и полгигабайта оперативной памяти для поддержки пользовательского интерфейса. В результате вся вычислительная мощь может быть задействована для решения насущных задач, а не тех, до которых вам нет дела. Если пользователь выберет приятный графический интерфейс с необычными прибамбасами, FreeBSD его поддержит, но не станет требовать, чтобы он работал только с ним.

Упрощенное управление программным обеспечением

Благодаря коллекции «портов» система FreeBSD облегчает управление программным обеспечением. Традиционно в системе UNIX настройка программного обеспечения требовала значительной квалификации. Коллекция «портов» существенно упрощает эту задачу за счет автоматизации и документирования установки, удаления и конфигурирования тысяч программных пакетов. (В некоторых других операционных системах BSD реализована своя система пакетов, основанная на коллекции «портов».)

Оптимизированный процесс обновления

В отличие от таких ОС, как Windows, в которых процедура обновления мучительна и опасна, простой процесс обновления FreeBSD строит операционную систему, оптимизированную для работы на конкретной программно-аппаратной платформе. FreeBSD учитывает каждую ее особенность, а не сводит поддержку к наименьшему общему знаменателю. Заменяя аппаратные средства, можно пересобрать систему с учетом их особенностей. Именно так поступают Sun и Apple, поскольку они создают как аппаратные средства, так и операционные системы. В то же время FreeBSD не привязывает вас к конкретной аппаратной платформе.

Файловая система

Файловая система (*filesystem*) определяет, как информация хранится на физическом диске. Выражение «моя веб-страница» преобразуется в последовательность нулей и единиц на металлической пластине жесткого диска. В состав FreeBSD входят очень сложные файловые системы. Они могут поддерживать файлы размером до петабайта (тысяча тысяч гигабайт). Они обладают высокой устойчивостью к сбоям, а запись и чтение файлов происходят чрезвычайно быстро. Файловая система BSD настолько усовершенствована, что принята многими производителями коммерческих UNIX, например компаниями Sun и HP.

Кому следует применять FreeBSD

FreeBSD может применяться как очень мощная настольная система и среда разработки, но ее история свидетельствует о преимущественной ориентации на Сеть, почтовые и файловые службы, а также сервисы поддержки. По существу, основная сильная сторона FreeBSD – это серверы Интернета. Система FreeBSD представляет собою замечательный выбор для любого сервиса Интернета.

Тому, кто задумывается о запуске FreeBSD (или любой другой системы UNIX) на настольной машине, необходимо понимать, как работает компьютер. FreeBSD – это не лучший выбор, если вы ищете простую систему с «мышиным» интерфейсом. Если ваша цель такова, выберите компьютер Макинтош и Mac OS X, которая базируется на BSD; следовательно, при необходимости вы сможете получить доступ к мощи UNIX и не вспоминать о ней уже никогда. Тот же, кто ищет наименьший общий знаменатель, всегда может выбрать из различных вариантов Microsoft Windows. Ему не потребуется понимать работу компьютера, ведь Windows – система легкая.

FreeBSD как настольная среда

Конечно, вы можете применять FreeBSD как мощную настольную ОС.

В компьютерном мире широко известно понятие, формулируемое как «eating your own dog food» (поедание собачьего корма). Если бы вы управляли

компанией, производящей собачий корм, то старались бы производить продукты, от которых не отказалась бы и ваша собака. А если она отводит нос от вашего последнего продукта, значит, у компании есть трудности. Суть такова: если вы создаете продукт, вы должны его использовать.

Самым быстрым методом обучения является метод полного погружения. Именно этот подход я избрал при изучении UNIX. Я работал с FreeBSD только на своем настольном компьютере и учился заставлять UNIX решать мои задачи. В результате я стал намного более сильным администратором.

На своем лэптопе с FreeBSD я даже написал книгу при помощи текстового редактора с открытым кодом (Emacs) и бизнес-пакета StarOffice. Кроме того, во FreeBSD я смотрю MPEG-видео с незашифрованных видео-CD и DVD, записываю компакт-диски с MP3 на собственном приводе и слушаю MP3 вместо того, чтобы работать. Это действительно исчерпывающий список настольных задач.

Настольные операционные системы также позволяют выполнять и дурацкие задачи. Сейчас в моей системе работает маленький анимированный демон BSD, спящий под указателем мыши. Когда я перемещаю мышь, демон просыпается, догоняет указатель и наносит удар своими вилами. Это ли не дурацкая задача?

Кто должен запускать другую BSD

Ближайшим конкурентом FreeBSD является NetBSD. Однако, в отличие от конкурентов в коммерческом мире, здесь конкуренция большей частью дружелюбна. NetBSD и FreeBSD свободно разделяют код и разработчиков; некоторые разработчики даже поддерживают одну и ту же подсистему в обеих операционных системах. Например, NetBSD и FreeBSD имеют общий код поддержки USB. В действительности на момент написания этих строк ведутся активные работы по интеграции сервера FTP, используемого в обеих операционных системах.

Основное преимущество NetBSD заключается в том, что она работает где угодно. Например, у меня есть древняя рабочая станция Silicon Graphics, на которой запущена NetBSD. Я использую эту машину в качестве сервера NFS (Network File System) и DNS (Domain Name System). И она справляется с работой. Если у вас есть старые или необычные аппаратные средства, NetBSD будет хорошим выбором.

Кажется, что OpenBSD стоит в стороне от остальных проектов BSD. Хотя ее код доступен для широкого использования, разработчики, похоже, больше заинтересованы в безопасности, а не в доступности своей системы. Однако в OpenBSD есть функции, которые облегчают выполнение таких задач, как создание мостов и брандмауэров, поэтому если вы не сможете обеспечить безопасную работу в FreeBSD, обратитесь к OpenBSD.

Кто должен запускать патентованную операционную систему

Патентованные операционные системы, подобные Solaris компании Sun, Windows NT компании Microsoft, AIX компании IBM, и другие системы такого рода по-прежнему довольно популярны, хотя разновидности BSD и Linux расширяют свое присутствие на рынке. Особенно следует отметить Solaris, которая удерживает значительную часть рынка UNIX.

Ведущие предприятия (из списка Fortune 500) сильно привязаны к Solaris и Windows NT. Тенденция немного меняется, но сейчас положение остается именно таким, и в подобных компаниях, вероятно, по-прежнему работают с этими операционными системами. Однако применение FreeBSD для обслуживания базовых сервисов и файлов может сделать вашу жизнь намного легче, потребовав за это намного меньше денег.

Конечно, если ваше программное обеспечение работает только на патентованной системе UNIX, то выбор операционной системы ясен. Тем не менее всегда спрашивайте производителя программ, доступна ли версия для FreeBSD; вас может ждать приятный сюрприз.

Как читать эту книгу

Толщина многих компьютерных книг достаточна для того, чтобы ими можно было оглушить быка, – если вы сможете поднять их достаточно высоко без бандажа и спинного корсета. Кроме того, они либо всеобъемлющи, либо настолько насыщены деталями, что их трудно читать. Нужен ли вам снимок экрана, который говорит только «нажмите OK» или «примите лицензионное соглашение»? Когда последний раз вы читали энциклопедию?

«FreeBSD. Подробное руководство» немного другая. Она предназначена для однократного прочтения от начала и до конца. При желании главы можно пропускать, однако каждая глава построена на материале предыдущей. Кроме того, книга не очень большая, поэтому усвоить ее содержимое довольно легко. Прочитав ее один раз, вы сможете применять ее в качестве справочника.

(Если вы часто покупаете компьютерные книги, можете добавить обычные рекомендации типа «для наилучшего усваивания читайте за раз одну главу» и т. д. Я не собираюсь няньчиться с вами, – если вы приобрели книгу о компьютерах, то, возможно, пара извилин у вас найдется. Внимательно работайте с примерами, и вы научитесь.)

Что вы должны знать?

Эта книга адресована начинающим администраторам UNIX. Несколько лет назад начинающий администратор UNIX уже был квалифицированным пользователем UNIX, обладающим навыками программирования и степе-

нью по вычислительной технике или по крайней мере чем-то одним. Сегодня операционные системы, подобные UNIX, свободно доступны в Интернете. Даже 12-летние дети могут запускать UNIX, читать исходный код. Они способны настолько хорошо изучить систему, что наводят ужас на нас, старых администраторов. Таким образом, для того чтобы приступить к чтению, глубокое знание UNIX не требуется.

Чтобы использовать весь потенциал этой книги, вы должны быть знакомы с базовыми командами UNIX. Это команды для перехода в другой каталог (cd), вывода списка файлов в каталоге (ls), а также вход в систему с именем пользователя и паролем. Тем, кто не знаком с базовыми командами и запуском UNIX из командной оболочки, я рекомендую начать с книги Эви Немет (Evi Nemeth), Гарта Снайдера (Garth Snyder), Скотта Сибасса (Scott Seebass) и Трента Р. Хейна (Trent R. Hein) «UNIX System Administration Handbook»[1], Prentice Hall.

Кроме того, надо немного разбираться в аппаратных средствах PC. (Не очень глубоко, уверяю вас, но в некоторой степени.) Например, полезно знать, что такое IRQ (interrupt request, запрос прерывания) и чем различаются жесткие диски SCSI и IDE. Конечно, надо знать именно свои аппаратные средства, однако если вы заинтересовались этой книгой, то такими знаниями вы обладаете. Задача немного облегчится, если предположить, что компьютер будет выделен исключительно для FreeBSD; в конце концов, многие сетевые серверы могут загружать как Windows, так и FreeBSD!

Примечание

«FreeBSD. Подробное руководство» рассказывает об администрировании FreeBSD, а не о том, как перенаправить вывод команды оболочки. Однако для того чтобы помочь начинающим администраторам, я включил в книгу команды оболочки, необходимые для достижения желаемых результатов. Те, кто лучше всего учится на примерах, найдут в этой книге все необходимое.

В наши дни многие начинающие системные администраторы обладают опытом работы в Windows. Они узнают, что «ls» подобна «dir», а «cd» одинаково выполняется на обеих платформах. Команды можно заучивать наизусть, читать их описания и исследовать на практике. Однако таким образом нельзя понять, как думает машина UNIX. Она не приноровится к вам; вы должны приспособить ее под себя. Помня об этом, мы потратим некоторое время на обсуждение того, как следует думать о системе FreeBSD.

Как думать о UNIX

Тот, кто собирается работать с FreeBSD, должен понимать некоторые способы мышления, принятые в мире UNIX. Пользователи с опытом работы в

[1] Э. Немет, Г. Снайдер, Т. Р. Хейн, С. Сибасс «UNIX. Руководство системного администратора», ISBN 5-7315-0021-5, «BHV-Киев», 2000 г.

Э. Немет, Г. Снайдер, Т. Р. Хейн, С. Сибасс «UNIX. Руководство системного администратора. Для профессионалов», ISBN 5-318-00754-6, «Питер», 2003 г.

Windows могут быть шокированы результатами первых попыток администрирования системы FreeBSD, если они не поймут, почему UNIX ведет себя именно так и какого поведения UNIX ожидает от них.

Пользователи, привыкшие к средам GUI, таким как Windows и Macintosh, наверняка не знакомы с тем, как UNIX обслуживает ввод и вывод. Если вы новичок в UNIX, то, возможно, привыкли нажимать на что-то и видеть либо сообщение «OK», либо ошибку, либо вообще ничего, либо (слишком часто) красивый голубой экран, заполненный терминами из области высоких технологий, объясняющими, где произошла авария в системе. В UNIX все немного по-другому.

Каналы коммуникации

Программы UNIX имеют три «канала»[1] коммуникации: стандартный ввод, стандартный вывод и стандартная ошибка. Поняв работу этих каналов, вы сделаете хороший шаг на пути к пониманию работы компьютера.

Стандартный ввод (*standard input*) – это источник информации. Когда вы набираете команды за консолью, стандартным вводом является клавиатура. Если программа прослушивает сеть, стандартным вводом считается сеть. Многие программы могут переназначать стандартный ввод, чтобы принимать данные из сети, файла, с клавиатуры и из любого другого источника.

Стандартный вывод (*standard output*) – это место, где отображается вывод программы. Зачастую это консоль (экран). Сетевые программы обычно возвращают вывод в сеть.

Наконец, *стандартный поток ошибок* (*standard error*) – это место, куда направляются сообщения об ошибках. Часто консольные программы возвращают ошибки на консоль; другие протоколируют ошибки в файле.

Работа с каналами

Описанные каналы можно переназначать произвольным образом, что, вероятно, труднее всего усваивается начинающими пользователями и администраторами UNIX. Такое переназначение кажется достаточно простым, но привыкнуть к нему немного труднее, чем кажется на первый взгляд.

Например, сообщения об ошибках можно перенаправить с терминала в файл. Если вы не хотите набирать список сведений в командной строке, можете поместить эту информацию в файл (ее можно будет использовать повторно) и выполнить дамп файла на стандартный ввод команды. Или еще лучше: запустите команду, чтобы сгенерировать эту информацию и поместить ее в файл или просто организовать конвейер и отправить информацию напрямую второй команде.

---

[1] В литературе по UNIX эти каналы чаще называют «потоками»: потоки ввода, вывода и ошибок. – *Примеч. науч. ред.*

Командная строка

Крайние проявления каналов ввода/вывода могут ошеломить нового пользователя. Когда при прохождении учебных курсов по администрированию UNIX я впервые увидел специалиста, набирающего в командной строке нечто вроде нижеследующего, я захотел сменить профессию.

```
# tail -f /var/log/messages | grep -v sudo | grep -v named &
```

Строки малопонятного текста начали появляться на экране. Хуже того, мой инструктор продолжал набирать, когда вывод закрыл собой весть экран!

Длинная строка команд определенно может запугать выходцев из среды с «мышиным» интерфейсом. Что означают все эти странные слова, не говоря уже о символах?

Изучение работы в командной строке можно рассматривать как изучение языка. Учась языку, мы начинаем с простых слов. По мере расширения нашего словаря мы узнаем, как связывать слова. Имея дело с командной строкой UNIX, вы начинаете с простых команд и только потом связываете их в чудовищные выражения, подобные показанным ранее.

Другая трудность – общий подход программных функций UNIX. В состав большинства коммерческих операционных систем входят монолитные программные пакеты, которые пытаются угодить всем, кто работает за компьютером. Программы UNIX – это простые, компактные инструменты. Частично это объясняется наличием перенаправляемых каналов ввода/вывода, частично – наследием UNIX. Вспомните, что когда-то работать с системой UNIX могли только программисты. А программисты ничего не имеют против того, чтобы создавать свои инструменты. Сборку утилитки в командной строке легко сравнить с компиляцией всего программного пакета.

Кроме того, эти маленькие программы предоставляют беспримерную гибкость. Хотелось ли вам когда-нибудь использовать функцию одной программы в другой? Применяя множество маленьких программ и назначая ввод и вывод по своему усмотрению, можно задавать любое поведение системы. Многие современные платформы восприняли эту идею маленьких, повторно используемых инструментов лишь в последние несколько лет.

Структура книги

«FreeBSD. Подробное руководство» включает следующие главы:

Глава 1 «Установка» содержит обзор установки FreeBSD на выделенной машине и советы по оптимальному конфигурированию.

Глава 2 «Как получить помощь» охватывает дополнительные информационные ресурсы, которые проект FreeBSD предоставляет пользователям. Ни одна книга не может охватить весь материал. Умение применять многие доступные ресурсы FreeBSB поможет вам заполнить любые пробелы в информации, которую вы здесь найдете.

Глава 3 «Прочтите это раньше, чем что-нибудь испортите! (Резервное копирование и восстановление)» описывает способы резервирования данных как на уровне всей системы, так и на уровне отдельных файлов. Здесь же рассказывается о том, как вносить изменения, которые легко отменить.

Глава 4 «Эксперименты с ядром» описывает конфигурирование ядра FreeBSD. В отличие от некоторых других операционных систем, ядро FreeBSD надо настраивать, чтобы оно лучше отвечало вашим целям. Такая возможность обеспечивает потрясающую гибкость и позволяет извлекать максимальную пользу из аппаратных средств.

Глава 5 «Работа в сети» содержит обсуждение протоколов TCP/IP и их работы во FreeBSD.

Глава 6 «Обновление FreeBSD» обучает вас, как задействовать потрясающий процесс обновления FreeBSD. Процедура обновления FreeBSD, по сравнению с тем, что предлагают любые другие операционные системы, – одна из самых замечательных и ровных.

Глава 7 «Организация защиты системы» научит вас, как защитить систему от взломщиков и злоумышленников.

Глава 8 «Расширенные средства защиты» описывает наиболее интересные функции защиты, такие как фильтрация пакетов и виртуальные машины.

Глава 9 «Каталог /etc» описывает основные конфигурационные файлы FreeBSD и их применение.

Глава 10 «Делаем систему полезной» описывает систему «портов» и пакетов, которая применяется для управления дополнительным программным обеспечением.

Глава 11 «Расширенное управление программным обеспечением» описывает управление системным программным обеспечением FreeBSD.

Глава 12 «Поиск хостов с помощью DNS» описывает систему доменных имен (Domain Name Service) и обучает отладке и установке сервиса имен.

Глава 13 «Управление малыми сетевыми сервисами» описывает инфраструктуру FreeBSD для поддержки сетевых сервисов и малых сетевых программ.

Глава 14 «Почтовые службы» обучает установке, применению и обслуживанию электронной почты как на сервере, так и на стороне клиента.

Глава 15 «Веб- и FTP-сервисы» объясняет, как запускать самые популярные службы Интернета.

Глава 16 «Файловые системы и диски» рассматривает детали файловых систем FreeBSD, поддержку других файловых систем и систему Soft Updates, которая делает FreeBSD столь надежной.

Глава 17 «RAID» охватывает встроенное программное обеспечение FreeBSD для управления массивами жестких дисков.

Глава 18 «Производительность системы» описывает некоторые инструменты FreeBSD для тестирования производительности и устранения неполадок.

Глава 19 «Как обстоят дела?» обучает мониторингу системы с помощью различных инструментов.

Глава 20 «Аварии и паника в системе» обучает, как поступать в редких случаях неполадок с FreeBSD, как выявлять и устранять проблемы и готовить отчет о проблеме.

Глава 21 «Настольная система FreeBSD» содержит указатели на множество различных программ, которые делают FreeBSD солидной и надежной платформой для рабочей станции.

Послесловие является кратким заключением по всему материалу.

Наконец, в приложении «Полезные sysctl MIB» описаны наиболее интересные и полезные опции настройки ядра.

Итак, довольно вводного материала! К установке!

1

Установка

Перед освоением системы FreeBSD ее надо установить. Для успешной инсталляции необходимо как программное обеспечение (FreeBSD), так и соответствующие аппаратные средства. FreeBSD можно достаточно легко получить, заглянув на *http://www.FreeBSD.org/* и щелкнув по ссылке «Getting FreeBSD». Также эту систему можно заказать у нескольких поставщиков, таких как FreeBSD Mall (*http://www.FreeBSDmall.com*) или Daemon News (*http://www.daemonnews.org*).[1]

Аппаратное обеспечение – совсем другое дело.

Аппаратное обеспечение FreeBSD

FreeBSD может работать на различных платформах, из которых наиболее популярны Intel-совместимые системы 80386 и более новые. Также FreeBSD можно запускать на последней и уже оплаканной Compaq Alpha. Кроме того, ведутся работы по переносу FreeBSD на платформы SPARC, StrongARM и PowerPC.

В этой книге обсуждается платформа Intel (известная еще как x386 и i386), поскольку системы на этой платформе наиболее распространены и поддерживаются лучше других. Вероятно, вы работаете на одной из них. FreeBSD может работать даже на старых системах. Возможно, у вас в кладовке найдется такая, которая прекрасно справится с этой задачей. Поскольку внимание в книге сосредоточено на сетевых серверах, приводимые инструкции относятся к установке FreeBSD на соответствующих машинах. Чтобы узнать, как наладить сосуществование FreeBSD с другими операционными системами, обратитесь к встроенной справочной документации FreeBSD.

В любом случае FreeBSD будет отлично работать на системах с определенной минимальной конфигурацией. Вот несколько основных рекомендаций.

---

[1] Я особенно рекомендую именно этих поставщиков, поскольку часть своей выручки они возвращают в сообщество FreeBSD. Можно найти поставщиков, которые предлагают более низкие цены, однако все деньги они оставляют себе.

Процессор

Для работы FreeBSD марка процессора несущественна. Неважно, будет ли это процессор Intel, AMD, IBM или Cyrix. Тип процессора выясняется при загрузке, а далее FreeBSD учитывает все особенности микросхемы, которые были распознаны. Когда-то серверы эффективно работали на машинах с процессорами i486 и даже переполняли каналы T1, предоставляющие доступ к Интернету. Однако все-таки лучше применять процессоры Pentium или более быстрые. Прогон некоторых примеров из этой книги займет менее часа при наличии двух процессоров Pentium 1 ГГц, а системе с процессором i486, работающим на частоте 25 МГц, для этого потребовалось бы почти три дня.

Память (RAM)

Прежде всего, память (RAM) – это хорошо, и чем больше памяти, тем лучше. Добавление микросхем RAM увеличит быстродействие системы заметнее, чем модернизация любых других компонентов. Абсолютный минимум RAM, необходимый системе, – 16 Мбайт.

Жесткие диски

Жесткие диски могут быть узким местом производительности. Несмотря на то что диски IDE очень дешевы, их производительность ниже, чем у дисков SCSI. Система SCSI может на полной скорости передавать данные между контроллером SCSI и каждым диском, тогда как скорость обмена данными с контроллером IDE тем меньше, чем больше дисков. Кроме того, контроллер SCSI может обслуживать до 15 дисков, а контроллер IDE – не больше двух. Пятнадцать дисков, работающих на полной скорости, или два диска с «поделенным» быстродействием – большая разница для пропускной способности!

Как бы то ни было, при наличии дисков IDE их лучше подсоединять к отдельным контроллерам. Сегодня во многих системах жесткий диск подсоединен к одному контроллеру IDE, а привод CD-ROM – к другому. Добавляя второй жесткий диск, подключите его ко второму контроллеру. (Как-никак, обращение к CD-ROM происходит реже, чем к жесткому диску.)

Наличие в системе 1 Гбайт дискового пространства вполне вас удовлетворит, но для работы с книгой необходимо не менее 10 Гбайт.

Загрузка FreeBSD

Если вместо покупки CD загружать FreeBSD по FTP, то полный список зеркал можно найти на *www.FreeBSD.org*. Впрочем, зеркала можно легко выбрать и без списка. Имена зеркальных серверов соответствуют следующему шаблону:

```
ftp<number>.<country>.FreeBSD.org
```

Замыкающий код страны может отсутствовать, и тогда под местом расположения сервера обычно подразумевается континентальная часть США, например *ftp3.FreeBSD.org*, *ftp2.uk.FreeBSD.org*, *ftp.ru.FreeBSD.org* и т. д.

Как правило, FTP-зеркала с меньшими номерами более загружены, чем серверы с большими номерами. Можно попробовать *ftp5.FreeBSD.org*[1] или сервер с высоким номером, имя которого включает в себя код страны, и посмотреть, насколько быстрым будет соединение.

Многие серверы, являющиеся зеркалами FreeBSD, содержат и другое программное обеспечение. На таких серверах FreeBSD находится в каталоге */pub/FreeBSD*. Вот как может выглядеть его содержимое:

```
..
.message
.notar
CERT
CTM
CVSup
FreeBSD-current
FreeBSD-stable
README.TXT
branches
development
dir.sizes
distfiles
doc
index.html
ls-lR.gz
ports
releases
snapshots
tools
updates
```

Сколько тут всего! К счастью, на большую часть файлов можно не обращать внимания. При первоначальной установке самым важным является каталог *releases/i386*. В нем можно найти полный перечень всех текущих версий FreeBSD, которые хранятся на зеркале, а также название каталога, в котором находятся ISO-образы для создания загрузочного компакт-диска. (См. справочную документацию для пишущего CD-ROM.)

Установка по FTP

Загрузка всего ISO-образа многим совершенно не нужна, поскольку ISO включает компоненты на самом деле бесполезные – десятки пакетов, которые, скорее всего, нет необходимости устанавливать. При хорошей пропускной способности канала (подразумевается кабельный модем, корпоративная локальная сеть или надежная линия 56 Кбит/с и наличие времени) лучше установить FreeBSD по FTP.

---

[1] Поскольку здесь упомянуто конкретное имя FTP-сервера, он будет перегружен запросами читателей, которые неукоснительно следуют приводимым инструкциям. Попробуйте. Выберите любой сервер. Не прекращайте поиски, пока не найдете подходящее зеркало.

При установке через FTP прежде всего необходимо загрузить два образа гибких дисков и создать дискеты. Почему дискеты? Да, загрузка с дискет отнимает время – по современным стандартам устройства чтения-записи гибких дисков считаются медленными. Однако они присутствуют в большинстве систем и обычно работают без сучка, без задоринки. Такие образы гибких дисков подобны старым добрым загрузочным дискам DOS; они содержат достаточную информацию для загрузки FreeBSD, запускают программу установки, считывают данные с CD или с FTP-сервера и записывают их на диск.

Образы гибких дисков можно найти в каталоге, соответствующем выбранной версии (например, 4.5-RELEASE), в подкаталоге *floppies*. Здесь представлены следующие файлы:

- **boot.flp** – образ для дисков 2,88 Мбайт. Полезен лишь при наличии устройства, работающего с гибкими дисками 2,88 Мбайт.
- **fixit.flp** – этот диск хранит несколько основных утилит, необходимых для восстановления системы. Дополнительную информацию см. в главе 3.
- **kern.flp** – образ загрузочного диска. Содержит основное ядро и «разговаривает» с аппаратными средствами в процессе установки.
- **mfsroot.flp** – образ второго загрузочного диска. Содержит программы, необходимые для установки FreeBSD на файловые системы, расположенные в RAM на электронном диске.

Из описанных дисковых образов действительно необходимы только файлы *kern.flp* и *mfsroot.flp*. Их следует скопировать на дискеты. Главное – не применять копирование на уровне файловой системы, подобное типичной операции «перетащить и оставить» (drag and drop) в Windows. Это файлы-образы, поэтому их следует копировать особым образом.

В системе UNIX команда `dd` выполнит все, что необходимо. Однако прежде всего надо выяснить имя устройства чтения/записи гибких дисков. Вероятно, это */dev/fd0*, */dev/floppy* или */dev/rfd0*.

Если имя устройства */dev/fd0*, то для записи образа *kern.flp* на диск следует ввести:

```
# dd if=kern.flp of=/dev/fd0
```

Для копирования файла *mfsroot.flp* на второй гибкий диск эту операцию необходимо повторить, заменив при наборе «kern.flp» на «mfsroot.flp».

При работе в Microsoft Windows нужна специальная утилита, позволяющая копировать образы дисков. Microsoft ее не предоставляет, однако она есть в поставке FreeBSD. Эту программу можно найти в подкаталоге *tools* основного каталога, соответствующего выбранной версии FreeBSD. Утилита называется «fdimage.exe».

Fdimage.exe – это свободно распространяемая программа для копирования дисковых образов, которую можно запускать под Windows. Применять ее довольно легко. Например, для копирования образа *kern.flp* на дискету в устройстве «a:» в строке приглашения DOS следует ввести:

```
c:> fdimage kern.flp a:
```

После того как дисковод завершит операцию (она займет какое-то время), этот процесс необходимо повторить для файла *mfsroot.flp* и второго жесткого диска.

Дополнительная информация для установки через FTP

Если в локальной сети для назначения IP-адресов и выполнения других операций применяется протокол динамической конфигурации хоста (Dynamic Host Configuration Protocol, DHCP), то Все Должно Работать. Однако если сетевой администратор назначает IP-адреса вручную, то у него необходимо получить следующую информацию:

- IP-адрес устанавливаемой системы FreeBSD
- IP-адреса серверов имен, присутствующих в сети
- Сетевой шлюз по умолчанию

Настройка аппаратных средств

Для настройки аппаратных средств придется совершить небольшое путешествие в BIOS. В большинстве компьютеров на экран настройки BIOS можно выйти непосредственно при загрузке. Обычно для этого надо нажать клавишу F2 или Delete.

В экране настройки необходимо выбрать носитель, с которого будет загружаться компьютер – либо гибкий диск, либо CD. Гибкие диски показаны как «floppy» или «A:». Устройства CD-ROM обычно представлены как «CD-ROM».

Примечание

Для получения помощи в настройке BIOS следует обратиться к справочному руководству или посетить сайт производителя BIOS.

Настраивая BIOS, опции «Plug and Play OS» следует присвоить значение «no». Такое значение предписывает BIOS выполнять основную настройку аппаратных средств, а не рассчитывать на то, что всю установку проведет операционная система. Современные версии Microsoft Windows предполагают, что аппаратные средства будут действовать согласно получаемым указаниям, и потому ожидают полный доступ к оборудованию. А FreeBSD предполагает, что система будет работать согласно стандартам аппаратных средств и требованиям спецификаций и потому использует преимущества настройки, которая легко выполняется в BIOS.

Примечание

Многие устройства (особенно сетевые карты) могут работать плохо, если не изменить эту опцию.

Непосредственная установка FreeBSD

При наличии загрузочного CD-ROM или двух загрузочных гибких дисков можно перезагрузить машину с любого из этих носителей. При загрузке должно появиться сообщение, предлагающее выбрать продолжение установки, конфигурирование ядра в визуальном меню или конфигурирование ядра в текстовом меню (рис. 1.1).

Рис. 1.1. *Первое меню загрузки*

В случае устаревших аппаратных средств необходимо настроить ядро, «рассказав» ему об имеющемся оборудовании. Например, FreeBSD поддерживает сетевые карты ISA с начала 1990-х, однако для надлежащей работы их необходимо тщательно конфигурировать. (Это ограничение аппаратного обеспечения, а не FreeBSD.)

Если карты ISA нет, то можно продолжить установку, однако при ее наличии следует настроить ядро для ее поддержки. По возможности необходимо заменить карты ISA на PCI – ими легче управлять и у них лучше пропускная способность. Однако при запуске FreeBSD на очень старых системах такая замена не всегда возможна.

Примечание

Для получения помощи при возникновении трудностей следует обратиться к Справочнику FreeBSD (FreeBSD Handbook), доступному в Сети по адресу *http://www.FreeBSD.org*. Если аппаратные средства произведены в последние несколько лет, установку можно продолжить без конфигурирования ядра.

Настройка ядра для поддержки карт ISA

При наличии карт ISA необходимо знать порядок определения и регулирования запросов на прерывания (Interrupt Requests, IRQ) и адресов портов памяти, которые жестко устанавливаются перемычками. (В противном слу-

чае порядок необходимо выяснить или, еще лучше, заменить эти карты.) После указания в FreeBSD надлежащей линии прерывания и адреса памяти карта будет работать.

В исходном меню загрузки (см. рис. 1) выберите конфигурирование ядра в полноэкранном визуальном режиме (full-screen visual mode). После этого откроется меню, представленное на рис. 1.2.

```
---Active-Drivers---------------------------? conflicts-------Dev---IRQ--Port--
  Storage :            (Collapsed)
  Network :            (Collapsed)
  Communications :     (Collapsed)
  Input :              (Collapsed)
  Multimedia :
  Miscellaneous :      (Collapsed)

---Inactive-Drivers-------------------------------------------Dev--------------
  Storage :
  Network :
  Communications :     (Collapsed)
  Input :
  Multimedia :
  Miscellaneous :      (Collapsed)
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
  [Enter] Expand device list      [X]    Expand all lists
  [TAB]   Change fields           [Q]    Save and Exit            [?] Help
```

Рис. 1.2. *Конфигурирование карты ISA*

Устройства разбиты на приблизительные категории: устройства хранения, сетевые и т. д. Выберите категорию, к которой принадлежит карта ISA, и нажмите Enter, чтобы раскрыть этот пункт. Если устройство не представлено в списке, то либо оно не требует конфигурирования, либо FreeBSD его не поддерживает. Необходимо выбрать устройство, ввести номер прерывания, закрепленный за картой, номер порта и адрес памяти в предоставляемом пространстве.

После того как ядро FreeBSD получит всю информацию о карте, надо нажать клавишу Q. Появится вопрос: следует ли сохранить конфигурацию и выйти из экрана настройки. Для продолжения следует нажать клавишу Y, в результате чего вы попадете в sysinstall.

Sysinstall: безобразный инсталлятор FreeBSD

Инсталлятор FreeBSD, показанный на рис. 1.3, – это печально известная система sysinstall, управляемая в режиме меню. В то время как другие операционные системы имеют симпатичные графические инсталляторы с меню, управляемыми мышью, и многоцветными секторными диаграммами, FreeBSD выглядит, как старая программа DOS. Даже автор системы отзывается о базовой библиотеке, как о «действительно ужасной». (Работа по ее замене ведется, но сейчас, когда я пишу эти строки, мне кажется, что sysinstall еще некоторое время из FreeBSD никуда не денется.)

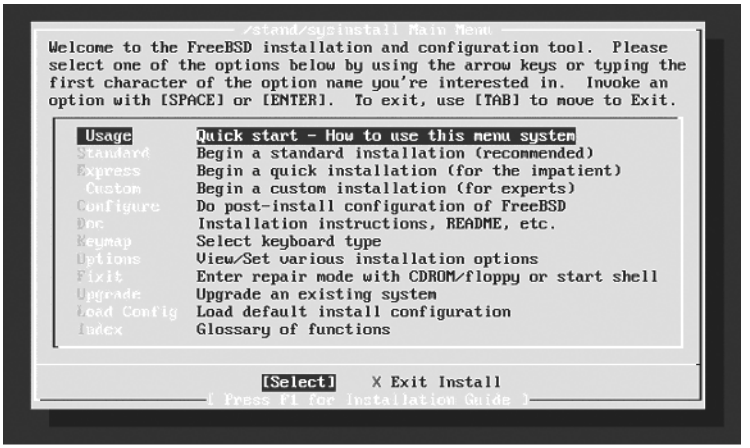

Рис. 1.3. *Основное меню sysinstal*

Вопреки первому впечатлению о sysinstall эту утилиту достаточно легко использовать, и работает она хорошо. Пошаговый разбор интерфейса здесь не представлен (в этом нет необходимости), но далее описываются различные пункты, которые будут рассмотрены в ходе установки. Благодаря этому выбор можно будет делать осмысленно.

Вот очень важное указание при работе с sysinstall: *для выбора использовать клавишу пробела*. Забавно, что даже при наличии подсказки на нескольких экранах sysinstall, в справочных файлах и инструкциях люди упускают это из виду. Затем, когда на это обратят их внимание, остаток своих дней они буду удивляться, как они могли это упустить. Если при выборе необходимого пункта не нажать пробел, установка закончится неудачей.

Да, и еще: на первый раз выберите стандартную установку (Standard install). Опуститесь на одну строку и нажмите Enter.

Использование диска

Многие пользователи настраивают компьютер так, чтобы можно было загружать несколько операционных систем. Они разбивают жесткие диски на разделы исходя из ОС, резервируя один участок для Windows, а оставшуюся часть – для других операционных систем. При такой настройке FreeBSD работает замечательно. Однако, поскольку предполагается создавать сервер Интернета, следует отвести под FreeBSD все дисковое пространство. Серверы Интернета должны быть активны все время, и никто не будет останавливать почтовый сервер компании, чтобы поиграть в Цивилизацию на разделе Windows!

При стандартной установке FreeBSD в начале и в конце диска остаются крошечные разделы, помечаемые как «неиспользуемые» (unused). Пустое пространство остается при форматировании диска для любых операционных систем, просто FreeBSD показывает расположение этих разделов. (Как обычно, архив почтовой рассылки, пополняемой FreeBSD-хакерами, содер-

жит обременительный объем детальной информации, которая объясняет причины такой разбивки.)

Примечание

При установке FreeBSD 3.X или предшествующих версий можно увидеть пункт «особо опасного» режима («dangerously dedicated» mode). Выбор этого режима устранит создание крошечных разделов в начале и конце диска. Некоторые очень-очень старые жесткие диски лучше всего работают именно в этом режиме. «Особо опасный» режим следует применять лишь тогда, когда предполагается его исследование или есть готовность иметь дело с последствиями. Поиск в архивах почтовой рассылки FreeBSD-hackers, расположенных по адресу *http://www.FreeBSD.org/*, – хороший источник информации по этой теме (см. главу 2).

Вначале инсталлятор представит список всех разделов диска. Их надо удалить, опускаясь вниз и нажимая клавишу D. В примере, представленном на рис. 1.4, показаны четыре раздела: два «неиспользуемых» раздела, рассмотренных ранее, один раздел FAT (для Windows) и один расширенный раздел Windows (extended).

```
Disk name:      ad0                                   FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

         0         63         62        -      6     unused        0
        63    4193217    4193279    ad0s1      2        fat       14    >
   4193280       1008    4194287        -      6     unused        0    >
   4194288   12319776   16514063    ad0s2      4   extended       15    >



The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable
T = Change Type        U = Undo All Changes     Q = Quit

Use F1 or ? to get more help, arrow keys to select.
```

Рис. 1.4. *Fdisk с разделами Windows*

Удалив все старые разделы, нажмите клавишу A для закрепления всего дискового пространства за FreeBSD. Итоговый экран будет выглядеть примерно так, как на рис. 1.5.

Нажмите Q для выхода. Далее инсталлятор представит экран Boot Manager (Менеджер загрузки), показанный на рис. 1.6. Установите главную загрузочную запись (master boot record, MBR). При этом будет удален любой существующий менеджер загрузки, который применялся на компьютере для загрузки нескольких операционных систем. (Речь идет о создании сервера Интернета. Значит, дисковое пространство не будет разделяться, скажем, с Windows Me.) Просто опуститесь до пункта «Standard», нажмите клавишу пробела и далее Enter для выхода из экрана.

После этого инсталлятор представит меню Disklabel (метки дисков).

```
Disk name:      ad0                                   FDISK Partition Editor
DISK Geometry:  16383 cyls/16 heads/63 sectors = 16514064 sectors (8063MB)

Offset       Size(ST)        End      Name  PType       Desc  Subtype    Flags

     0          63           62         -      6     unused        0
    63    16514001     16514063     ad0s1      3     freebsd      165    CA



The following commands are supported (in upper or lower case):

A = Use Entire Disk    G = set Drive Geometry   C = Create Slice
D = Delete Slice       Z = Toggle Size Units    S = Set Bootable
T = Change Type        U = Undo All Changes     Q = Quit

Use F1 or ? to get more help, arrow keys to select.
```

Рис. 1.5. *Fdisk с одним разделом FreeBSD*

```
Install Boot Manager for drive ad0:
FreeBSD comes with a boot selector that allows you to easily
select between FreeBSD and any other operating systems on your machine
at boot time.  If you have more than one drive and want to boot
from the second one, the boot selector will also make it possible
to do so (limitations in the PC BIOS usually prevent this otherwise).
If you do not want a boot selector, or wish to replace an existing
one, select "standard".  If you would prefer your Master Boot
Record to remain untouched then select "None".

  NOTE:  PC-DOS users will almost certainly require "None"!

        BootMgr   Install the FreeBSD Boot Manager
        Standard  Install a standard MBR (no boot manager)
        None      Leave the Master Boot Record untouched

                [ OK ]        Cancel
{ Press F1 to read about drive setup }
```

Рис. 1.6. *Окно выбора Boot Manager*

Разбиение диска

Мы подошли к первой сложной части – разбиению жесткого диска. В отличие от других операционных систем, для которых характерна быстрая поверхностная разбивка на несколько разделов, FreeBSD позволяет задавать расположение каждого раздела на жестком диске. Это важно по нескольким причинам.

Каждый раздел доступен пользователю как *точка монтирования* (*mount point*). В Windows каждый раздел имеет точку монтирования, связанную с буквой (например, C: или D:). В UNIX все разделы и диски представляют собой часть единого дерева каталогов. Раздел назначается каталогу и «монтируется» в него. Один раздел закрепляется за корнем дерева каталогов (/), а остальные назначаются произвольно. Например, при создании большого

веб-сервера на диске можно предусмотреть раздел, предназначенный для веб-сайтов, и монтировать его как */www*. Далее мы поговорим о том, как назначать точки монтирования и разбивать жесткий диск.

Обычно FreeBSD разбивает диск на четыре основных раздела: / (или *root*), пространство свопинга (swap space), */var* и */usr*. Читатель, имеющий опыт работы с UNIX, возможно, захочет создать дополнительные разделы и назначить точки монтирования по своему усмотрению. Далее обсуждаются все основные разделы, а также высказываются некоторые соображения об их размере и расположении.

Примечание

Здесь и далее будут упоминаться другие разделы, которые можно создать. Читатель, не знакомый с ними, может их пропустить. Более опытные администраторы могут принять или проигнорировать этот совет по своему выбору.

Прежде всего, следует отметить, что внешний край диска движется быстрее внутреннего, поэтому чем ближе файл к краю диска, тем быстрее можно получить к нему доступ или изменить его. Наиболее важные файлы следует размещать ближе к краю, чтобы скорость их чтения и записи была больше. А менее динамичные файлы лучше размещать ближе к центру. (Доступ к данным, расположенным на вращающемся жестком диске, очень напоминает карусель – можно сидеть в середине и почти ничего не чувствовать, однако если разместиться ближе к краю и высунуть голову, то через несколько минут можно потерять устойчивость.[1])

Создание разделов на диске выполняется в меню Disklabel (рис. 1.7).

```
                    FreeBSD Disklabel Editor

Disk: ad0       Partition name: ad0s1   Free: 16514001 blocks (8063MB)

Part      Mount          Size Newfs   Part      Mount          Size Newfs
----      -----          ---- -----   ----      -----          ---- -----



The following commands are valid here (upper or lower case):
C = Create         D = Delete    M = Mount pt.
N = Newfs Opts     Q = Finish    S = Toggle SoftUpdates
T = Toggle Newfs   U = Undo      A = Auto Defaults

Use F1 or ? to get more help, arrow keys to select.
```

Рис. 1.7. *Меню Disklabel*

[1] Хорошо, большая часть этих ошеломляющих ощущений связана с центробежной силой, однако внешний край преодолевает в минуту большее расстояние, нежели внутренний. А теперь оставим в стороне эту аналогию.

Примечание

Разделы, представленные в меню разбивки диска ближе к верху, располагаются ближе к краю диска. Если требуется разбить несколько дисков, то это следует делать сразу, чтобы удержать в памяти порядок их следования.

Рассмотрим каждый стандартный раздел по очереди. Разделы нельзя изменять на работающей системе, поэтому важно правильно создать их с самого начала.

root

В системе необходим быстрый доступ к корневой файловой системе[1] (/), которая содержит ядро, а также утилиты и программы, достаточные для загрузки компьютера в основное начальное состояние – в однопользовательский режим (объяснено в главе 3). Поэтому раздел root следует разместить на внешнем крае диска и выделить для него не менее 128 Мбайт независимо от емкости диска.

В некоторых UNIX-подобных операционных системах, например в части дистрибутивов Linux, следует предусмотреть большой раздел root, который должен вмещать большую часть операционной системы, либо организовать один большой раздел root для всех файлов на диске. Эта идея плоха в силу различных причин. Во-первых, такой вариант не позволит управлять размещением файлов на диске. Это снизит производительность. Во-вторых, при повреждении диска систему с большей вероятностью можно загрузить, если раздел root мал. В этом случае появляется небольшой шанс восстановить данные.

Под раздел root надо отвести около 128 Мбайт. Нажмите c для создания раздела, наберите `128M` и нажмите Enter. Инсталлятор спросит, что создается: файловая система или пространство свопинга. Выберите «filesystem». Далее последует вопрос о точке монтирования. Наберите / и нажмите Enter.

Пространство свопинга

Следующий шаг подразумевает создание пространства свопинга – дискового раздела, используемого подсистемой виртуальной памяти. Когда физическая память заполнена, система перемещает в раздел свопинга информацию, которая не была задействована некоторое время. Размещение пространства свопинга ближе к внешнему краю диска в известной степени улучшает производительность.

Итак, каков необходимый объем пространства свопинга? Это предмет долгих дискуссий системных администраторов. Короткий ответ – «зависит от

[1] В русскоязычной литературе распространены два наименования этой файловой системы: корневая и root. В этой книге для краткости принят второй вариант. Помните, что файловая система root – это то, что смонтировано в каталог /. Есть еще каталог /*root* – просто подкаталог файловой системы root. – *Примеч. науч. ред.*

системы». Здравые размышления подсказывают, что объем пространства свопинга должен по крайней мере вдвое превышать емкость физической памяти. Это неплохое правило, если понимать, что оно весьма обобщенное. Больший объем делу не повредит, а меньший – вполне, если в системе возникнет нехватка оперативной памяти. Система виртуальной памяти FreeBSD предполагает наличие пространства свопинга, объем которого по крайней мере вдвое превышает емкость физической памяти, и исходя из этого предположения делает определенный выбор и проводит оптимизацию.

В конце концов, при добавлении памяти трудно организовать дополнительное пространство свопинга, поскольку это дисковый раздел. Для этого надо изменить размер раздела, что всегда отчасти рискованно! Общее правило таково: необходимо создать пространство свопинга, объем которого по крайней мере вдвое превышает предполагаемую емкость памяти. Если в настоящее время в системе есть 128 Мбайт RAM, но ожидается увеличение емкости до 1 Гбайт, то для свопинга надо выделить 2 Гбайт.

Распределение пространства свопинга

При наличии нескольких дисков эффективность свопинга можно значительно улучшить, распределив его между несколькими дисками. Первый участок свопинга можно разместить на втором с краю разделе загрузочного диска (с разделом root), а остальные – на крайних разделах других дисков. (Такая схема хорошо работает при наличии до четырех разделов на четырех дисках; если создается более четырех разделов свопинга, последующие разделы будут использоваться также оптимально, как и первые четыре.)

Однако наибольшую пользу из разделения пространства свопинга позволяют извлечь диски SCSI. Что касается дисков IDE, то они должны размещаться на различных контроллерах. Следует помнить, что пропускная способность каждого IDE-контроллера распределяется между присоединенными к нему жесткими дисками. Если на одном и том же IDE-контроллере размещены два жестких диска, доступ к которым осуществляется одновременно, то быстродействие каждого диска снизится наполовину. Главное узкое место при использовании пространства свопинга – это пропускная способность, поэтому такая компоновка ничего не даст.

Если пространство свопинга распределяется между несколькими дисками, то создаваемые разделы должны иметь примерно одинаковый объем. Во FreeBSD предусмотрена оптимизация для четырех разделов свопинга.

Однако наличие четырех разделов свопинга приводит к конфликтным ситуациям. При отказе системы FreeBSD может записать копию образа физической памяти в раздел свопинга. Это позволит разработчику определить и устранить неполадки, вызвавшие отказ системы. Однако для выполнения дампа памяти размер по крайней мере одного раздела свопинга должен быть как минимум не меньше емкости физической памяти системы. При наличии четырех разделов свопинга, размер каждого из которых сопоставим с емкостью физической памяти системы, есть четырехкратный запас для свопинга данных физической памяти. Это большое пространство свопинга, осо-

бенно для современных систем. Его объем даже в два раза больше объема, предписываемого правилом «двойной емкости физической памяти». Можно не сомневаться, что дополнительное пространство свопинга не причинит вреда. Кроме того, сейчас дисковое пространство стоит совсем недорого. Если место для свопинга *действительно* необходимо, то его можно получить. Во всяком случае, если система непрерывно обращается к разделу свопинга, то можно приобрести дополнительные микросхемы RAM.

Решив, сколько места отводится для свопинга, надо создать раздел, нажав клавишу C. Следует задать необходимый размер – так, для раздела свопинга 1000 Мбайт введите значение 1000m. На вопрос инсталлятора о том, что создается: раздел свопинга или файловая система, следует ответить «Swap».

/var, /usr/ и /home

Следующий шаг – создание раздела */var*, который хранит быстро изменяющиеся данные, такие как файлы протоколов, базы данных, почтовые ящики и тому подобное. Если в системе много файлов протоколов или почтовых файлов, для этого раздела вполне может потребоваться 1 Гбайт или более. На маленьких серверах для этого раздела зачастую предусматривают 20% остающегося дискового пространства, а на почтовых – 70% и более.

Раздел */usr* хранит программы операционной системы, исходный код и другие подобные файлы. Часто многие администраторы выделяют оставшееся дисковое пространство под раздел */usr* – это наиболее распространенная стратегия.

Примечание

При создании веб-сервера, на котором у каждого веб-сайта есть свой пользователь и домашний каталог, выделение оставшегося дискового пространства под */usr* может быть не самой лучшей идеей. В этом случае 3 Гбайт для */usr* будет более чем достаточно – фактически при любом варианте. Оставшееся пространство можно назначить */home* – разделу для домашних каталогов пользователей. Тогда файлы пользователей будут отделены от системных, а скорость доступа к файлам по сути потеряет значение, если превысит определенный допустимый минимум.

После завершения этих операций меню Disklabel будет выглядеть примерно так, как показано на рис. 1.8.

Второй жесткий диск

Из наличия второго жесткого диска, качество которого сравнимо с качеством основного, можно извлечь пользу, если действовать правильно. Во-первых, как обсуждалось в разделе «Распределение пространства свопинга», пространство свопинга должно примыкать к внешнему краю диска. Оставшуюся часть диска следует отвести для пользовательских данных, отделяя их от операционной системы. Для этого оставшееся дисковое пространство следует назначить разделу, где будут храниться файлы, необходимые для

```
                    FreeBSD Disklabel Editor

Disk: ad0       Partition name: ad0s1   Free: 0 blocks (0MB)

Part      Mount          Size Newfs   Part      Mount          Size Newfs
----      -----          ---- -----   ----      -----          ---- -----
ad0s1a    /             100MB UFS Y
ad0s1b    swap          516MB SWAP
ad0s1e    /var          100MB UFS Y
ad0s1f    /usr         7347MB UFS Y

The following commands are valid here (upper or lower case):
C = Create         D = Delete    M = Mount pt.
N = Newfs Opts     Q = Finish    S = Toggle SoftUpdates
T = Toggle Newfs   U = Undo      A = Auto Defaults

Use F1 or ? to get more help, arrow keys to select.
```

Рис. 1.8. *Disklabel после разбиения на разделы*

операций, которые призван поддерживать сервер. Если это почтовый сервер, то на втором диске надо разместить */var* или */var/mail*, а если веб-сервер, то надо создать раздел */www* или */home*. Если это хост для протоколирования событий в сети, дисковое пространство второго диска следует назначить разделу */var/log*.

Вообще, отделение операционной системы от пользовательских данных повысит эффективность системы. Как и другие эмпирические правила, такая стратегия является спорной. Но ни один сисадмин не скажет, что это плохая или опасная идея, – несмотря на то что они могут бесконечно дискутировать о других вариантах использования дискового пространства.

Если заранее неизвестно, для каких задач будет применяться сервер, на втором диске надо создать раздел */usr*, а большую часть дискового пространства первого выделить для */var*.

Если второй диск намного медленнее основного системного, то лучше с ним и не связываться.[1] Мало того что его производительность не будет достаточной, наверняка он старше основного диска и менее надежен.

При запуске инсталлятор FreeBSD обнаружит все жесткие диски системы и предоставит возможность распределить пространство каждого из них.

Software Updates

Во FreeBSD есть несколько замечательных приемов, связанных с файловыми системами. Их общее название – *Soft Updates*. Более подробно они рассмотрены в главе 16. А пока надо просто принять на веру, что при установке лучше

---

[1] Если это как раз ваш случай, лучше прервать чтение и заменить старый диск. Эксплуатация однородных элементов системы с разными надежностью и быстродействием допустима только в экспериментальных целях, но никак не на рабочей машине. – *Примеч. науч. ред.*

их включить. Читатель, изучивший Software Updates и разочаровавшийся в них, может легко от них отказаться. Об этом еще пойдет речь в главе 16.

Спускаясь по меню вниз, для выбора каждого раздела следует нажимать клавишу S. В результате Soft Updates будут включены.

Размер блока

В этом разделе описываются варианты выбора, следование которым может снизить производительность системы. Тем, кто не знаком с FreeBSD, следует принять установки по умолчанию. Эти варианты предназначены для опытных администраторов UNIX, которые точно знают, что они делают.

Размер блока определяет минимальный размер файла. Файл, содержащий всего один символ, займет целый блок, даже если это будет лишь формальное заполнение блока. Точно так же, если файл превышает размер блока, он займет один блок и фрагмент другого. Каждый блок можно разделить на фрагменты, поэтому несколько файлов, размер которых слегка превышает размер блока, могут размещать «хвосты» в одном блоке.

Во FreeBSD размер блока по умолчанию равен 8 Кбайт. При создании большого раздела – скажем, 1 Гбайт и более – следует задать размер блока 16 Кбайт. В этом случае также надо изменить размер фрагмента. Файловая система FreeBSD (UFS, Unix File System) лучше всего работает с фрагментами, размер которых равен одной восьмой размера блока. Это могут быть блоки размером 16 384 байт и фрагменты размером 2048 байт.

Размер блока устанавливается с помощью программы newfs. В меню Disklabel, выбрав соответствующий раздел, надо нажать клавишу N для отображения всплывающего диалогового окна, представляющего ключи программы news. Для того чтобы задать размер блока равным 16 Кбайт и размер фрагмента равным 2 Кбайт, надо ввести:

```
newfs -f 2048 -b 16384
```

Варианты установки

В следующем меню можно выбрать, что устанавливать. Вариантов выбора довольно много, но здесь они представлены в упрощенной форме. При создании сервера Интернета надо выбрать Developer (Разработчик). Для настольного компьютера или экспериментальной машины общего назначения можно выбрать All (Все) – это показано на рис. 1.9. (Необходимо помнить о нажатии клавиши Space для выбора варианта и клавиши Enter для подтверждения выбора!)

Далее инсталлятор спросит, устанавливать ли коллекцию портов (Port Collection). Выберите Yes, даже если вы не знаете, о чем речь.

При установке системы X Window, или для краткости X, следует выбрать варианты, предполагающие полную установку X. (Намного проще все предусмотреть на этом этапе, чем добавлять компоненты впоследствии.) Но при наличии опыта работы с FreeBSD можно всегда вернуться назад и отказаться от компонентов, в которых нет необходимости.

```
Choose Distributions
As a convenience, we provide several "canned" distribution sets.
These select what we consider to be the most reasonable defaults for the
type of system in question.  If you would prefer to pick and choose the
list of distributions yourself, simply select "Custom".  You can also
pick a canned distribution set and then fine-tune it with the Custom item.

Choose an item by pressing [SPACE] or [ENTER].  When finished, choose the
Exit item or move to the OK button with [TAB].

<<< X Exit              Exit this menu (returning to previous)
    All                 All system sources, binaries and X Window System)
    Reset               Reset selected distribution list to nothing
[ ] 4 Developer         Full sources, binaries and doc but no games
[ ] 5 X-Developer       Same as above + X Window System
[ ] 6 Kern-Developer    Full binaries and doc, kernel sources only
[ ] 7 X-Kern-Developer  Same as above + X Window System
[ ] 8 User              Average user - binaries and doc only
  ↓(+)

                [ OK ]        Cancel
[ Press F1 for more information on these options. ]
```

Рис. 1.9. *Меню Distributions*

Носители для инсталляции

Как показано на рис. 1.10, возможны различные варианты выбора носителей для проведения установки (installation media). Наиболее популярны CD-ROM и FTP. При наличии компакт-диска с FreeBSD предпочесть следует его. Если такого диска нет, но есть действующее сетевое соединение, уста-

```
Choose Installation Media
FreeBSD can be installed from a variety of different installation
media, ranging from floppies to an Internet FTP server.  If you're
installing FreeBSD from a supported CD/DVD drive then this is generally
the best media to use if you have no overriding reason for using other
media.

1 CD/DVD        Install from a FreeBSD CD/DVD
2 FTP           Install from an FTP server
3 FTP Passive   Install from an FTP server through a firewall
4 HTTP          Install from an FTP server through a http proxy
5 DOS           Install from a DOS partition
6 NFS           Install over NFS
7 File System   Install from an existing filesystem
8 Floppy        Install from a floppy disk set
9 Tape          Install from SCSI or QIC tape
X Options       Go to the Options screen

                [ OK ]        Cancel
[ Press F1 for more information on the various media types ]
```

Рис. 1.10. *Меню Installation Media*

новку можно проводить через FTP. Возможно, это как раз тот вариант, который надо выбрать, если были созданы дискеты. Находясь за брандмауэром, при установке по FTP следует выбрать либо установку «through a firewall» (через брандмауэр), либо «through a http proxy» (через http-прокси). Перед выбором любого варианта установки через FTP следует убедиться, что сетевой кабель подключен должным образом.

Кроме того, FreeBSD можно установить с магнитной ленты, через NFS, с помощью нескольких десятков дискет или воспользоваться другими носителями. Читатель, выбирающий один из них, либо уже является сисадмином UNIX (NFS, магнитная лента), либо повредил голову, упав на каменную кладку, и ждет, когда добрые дяди в белых одеждах перенесут его в палату, обитую войлоком (установка с дискет).[1]

При установке через FTP система выдаст меню, из которого надо выбрать интерфейс. Выберите карту Ethernet.[2] Далее будут заданы вопросы об IP-адресах. Эта информация была собрана при подготовке установки через FTP.

Подтверждение

После выбора носителя для инсталляции программа sysinstall запросит подтверждение. После ответа «Yes» начнет вращаться жесткий диск, загорится индикатор привода CD-ROM и можно будет идти обедать.[3] По возвращении обнаружится, что основная часть установки завершена.

Наконец, появится диалоговое окно, в котором будет задан вопрос о желании провести настройку после инсталляции (post-install setup). Выберите «Yes».

Настройка после инсталляции

Меню конфигурирования (Configuration menu) для настройки FreeBSD после установки позволяет выбрать значения основных параметров системы (рис. 1.11). Все элементы конфигурации будут обсуждены позднее, но для более легкой жизни основные настройки лучше провести сейчас.

Пароль root

Для начала задайте пароль пользователя root. Если такого пароля нет, любой болван может войти в систему как root, не набрав никакого пароля. (А поскольку root обладает полной властью над аппаратным и программным обеспечением, это будет плохо.) Выберите третий пункт меню, Set the system

---

1 Автор намекает на психиатрическую больницу. – *Примеч. перев.*

2 Автор предполагает, что через коммутируемое соединение или выделенную линию 256 Кбит/с никто не будет устанавливать FreeBSD – это очень долго, хотя технически возможно. – *Примеч. науч. ред.*

3 Установка FreeBSD 5.0 на компьютер с процессором Pentium IV и быстрыми жестким диском и приводом CD-ROM занимает минут пять–семь. Обедать придется быстро. – *Примеч. науч. ред.*

```
FreeBSD Configuration Menu
If you've already installed FreeBSD, you may use this menu to customize
it somewhat to suit your particular configuration.  Most importantly,
you can use the Packages utility to load extra "3rd party"
software not provided in the base distributions.

X Exit              Exit this menu (returning to previous)
  Distributions     Install additional distribution sets
  Packages          Install pre-packaged software for FreeBSD
  Root Password     Set the system manager's password
  Fdisk             The disk Slice (PC-style partition) Editor
  Label             The disk Label editor
  User Management   Add user and group information
  Console           Customize system console behavior
  Time Zone         Set which time zone you're in
  Media             Change the installation media type
  Mouse             Configure your mouse
  Networking        Configure additional network services

                [ OK ]        Cancel
[ Press F1 for more information on these options ]
```

Рис. 1.11. *Конфигурирование после инсталляции*

manager's password (Установить пароль менеджера системы). Как показано на рис. 1.12, система попросит ввести пароль root дважды. Пароль пользователя root следует помнить, ибо его восстановление в случае потери прибавит хлопот.

```
FreeBSD Configuration Menu
If you've already installed FreeBSD, you may use this menu to customize
it somewhat to suit your particular configuration.  Most importantly,
you can use the Packages utility to load extra "3rd party"
software not provided in the base distributions.

X Exit              Exit this menu (returning to previous)
  Distributions     Install additional distribution sets
  Packages          Install pre-packaged software for FreeBSD
  Root Password     Set the system manager's password
  Fdisk             The disk Slice (PC-style partition) Editor
  Label             The disk Label editor
  User Management   Add user and group information
  Console           Customize system console behavior
  Time Zone         Set which time zone you're in
  Media             Change the installation media type
  Mouse             Configure your mouse
  Networking        Configure additional network services

                [ OK ]        Cancel
[ Press F1 for more information on these options ]
```

Рис. 1.12. *Установка пароля root*

Добавление пользователей

По возможности все операции следует проводить, входя в систему с правами обычного пользователя. Регистрация под именем root необходима лишь при проведении изменений в системе. Первое время это происходит часто, но затем все реже и реже. Однако для входа в систему в качестве обычного пользователя необходимо создать его учетную запись.

Для того чтобы добавить обычного пользователя в меню настройки, надо выбрать пункт User Management (Управление пользователями). Появится небольшое меню, предлагающее добавить нового пользователя, новую группу или вернуться в меню настройки. Если выбрать пункт User, появится экран, представленный на рис. 1.13.

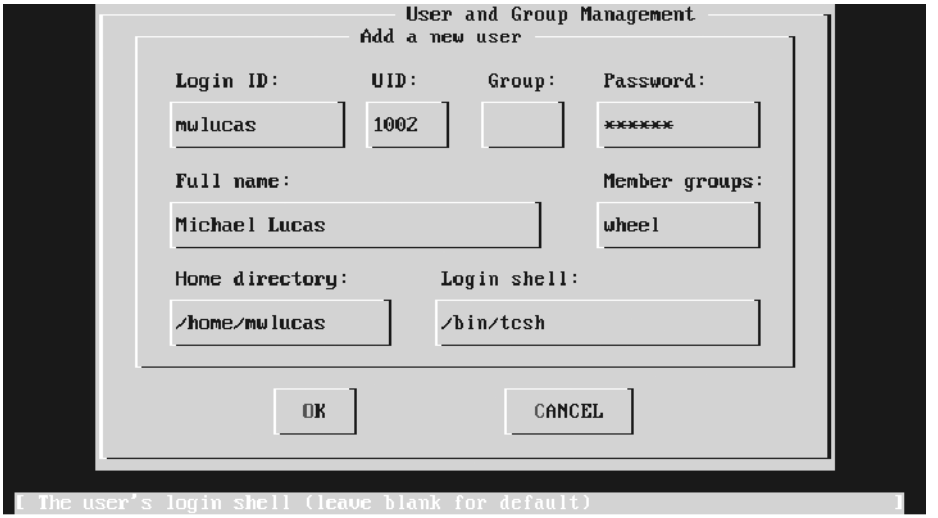

Рис. 1.13. *Добавление пользователя*

Прежде всего следует выбрать регистрационное имя-идентификатор (Login ID), или имя пользователя. Некоторые компании устанавливают стандарт, которому должны соответствовать имена пользователей. Это могут быть, например, инициалы и фамилия (инициалы помогают избежать неожиданных конфликтов).

Идентификатор пользователя (user ID, UID,) назначается системой. Опытный системный администратор может изменить его, но это не рекомендуется и обычно не приносит большой пользы.

По умолчанию во FreeBSD пользователь попадает в «группу» (Group), название которой соответствует имени пользователя. Например, пользователь «mwlucas» автоматически попадает в группу «mwlucas». Тот, кто знает, что он делает, может изменить это.

Full name (Полное имя) – это имя пользователя. Достаточно просто. Другие пользователи системы могут видеть это имя, поэтому не стоит выбирать его

произвольно. У некоторых системных администраторов возникали трудности, когда они назначали пользователям сомнительные имена, например «Pain in the Tuckus».

Home directory (Домашний каталог) – здесь хранятся файлы пользователя. Обычно подходит значение по умолчанию.

Member group (Группы пользователей) – простой список других групп системы, в которые будет входить пользователь. Если необходимо, чтобы пользователь мог применять пароль root и становиться пользователем root, этого пользователя следует добавить в группу «wheel». В этой группе должны присутствовать администраторы и отсутствовать пользователи. (Читателю следует убедиться, что его личная учетная запись подразумевает вхождение в группу wheel!)

Наконец, для нового пользователя надо выбрать командную оболочку (shell). Умудренные опытом администраторы и пожилые преподаватели зачастую предпочитают */bin/sh*. В примерах, приводимых в книге, подразумевается более дружелюбная командная оболочка */bin/tcsh*, входящая в современные дистрибутивы BSD.

После этого нажмите OK, и пользователь будет создан.

Часовой пояс

Для установки часового пояса выберите пункт Time Zone (Часовой пояс) в меню настройки (см. рис. 1.11). На вопрос, устанавливаются ли системные часы по универсальному времени (UTC), следует ответить «No» и перейти к представленным меню. Будут заданы вопросы, отвечая на которые, вы укажете континент, страну (рис. 1.14) и затем часовой пояс.

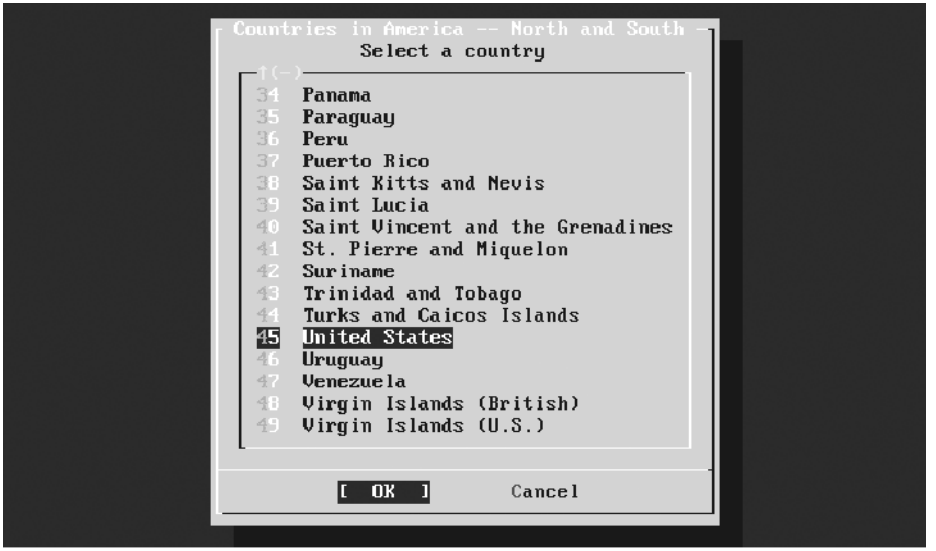

Рис. 1.14. *Выбор страны*

Мышь

Если в системе есть мышь, ее работу легко настроить сейчас. Если речь идет о старой доброй двух- или трехкнопочной мыши PS/2 или встроенном трекболле, в меню настройки надо выбрать пункт Mouse и затем Enable. Курсор мыши должен быть виден на экране и перемещаться при движении мыши.

При наличии мыши другого стандарта в меню, представленном на рис. 1.15, измените порт, с которым работает мышь, и ее тип.

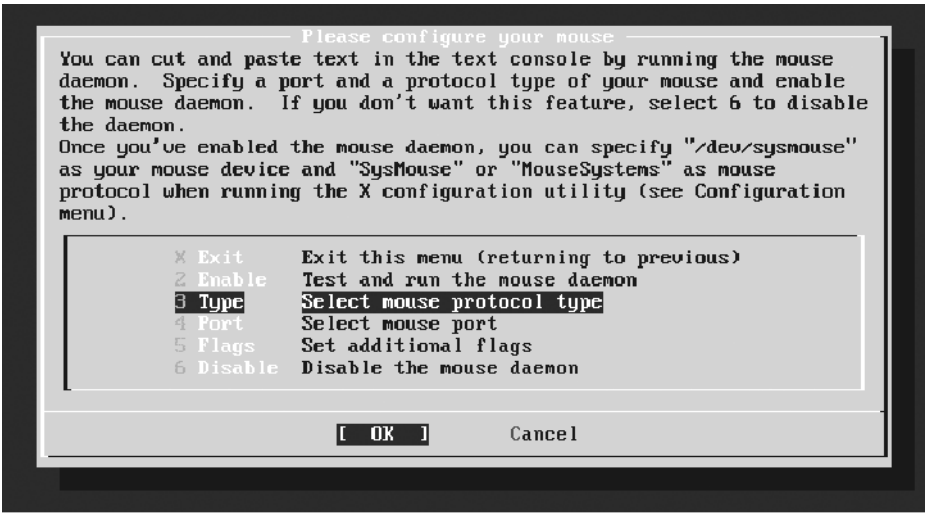

Рис. 1.15. *Конфигурирование мыши*

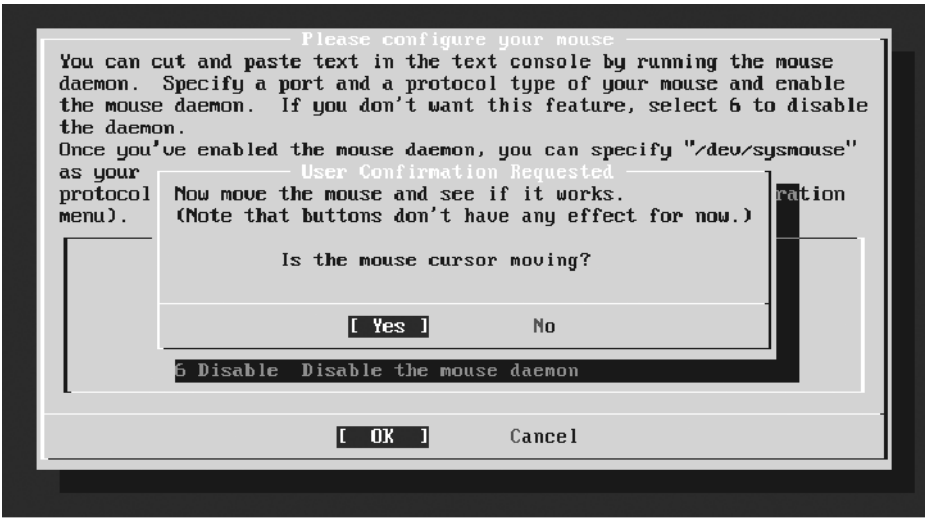

Рис. 1.16. *Меню тестирования мыши*

После установки типа мыши и порта выберите пункт Enable. Далее появится всплывающее меню, в котором будет задан вопрос, работает ли мышь. Подвигав мышью, выберите подходящий ответ – Yes или No (рис. 1.16). Если мышь не работает, значит, выбранные настройки ей не соответствуют. Встречаются мыши, работа которых не отвечает первоначальным представлениям о них. Для изменения настроек и повторной попытки выберите No.

Конфигурирование сетевых карт

Если в машине есть сетевая карта, а CD-ROM уже установлен, то сейчас, вероятно, надо сконфигурировать настройки сетевой карты. Это позволит избежать хлопот впоследствии. Карта должна быть соединена с сетью. В меню настройки выберите пункт Networking (Сеть), а затем пункт Interfaces (Интерфейсы) из меню Network Services (Сетевые службы), представленном на рис. 1.17.

Выберите интерфейсы, которые надо настраивать. (Новичок в компьютерных технологиях может даже не догадываться, что параллельный порт может быть сетевым интерфейсом!) Найдите пункт, в котором есть слово «Ethernet», и выберите его. На рис. 1.18 карта Ethernet называется «fxp0».

В появившемся диалоговом окне будет задан вопрос, следует ли конфигурировать интерфейс в соответствии с Ipv6. Если читатель не знает, что это такое, этот пункт нужно пропустить. Во втором окне будет предложено конфигурировать интерфейс по протоколу DHCP. Если в сети работает DHCP, такую настройку можно попробовать; в противном случае следует выбрать No. Появится экран настройки сетевого интерфейса, представленный на рис. 1.19.

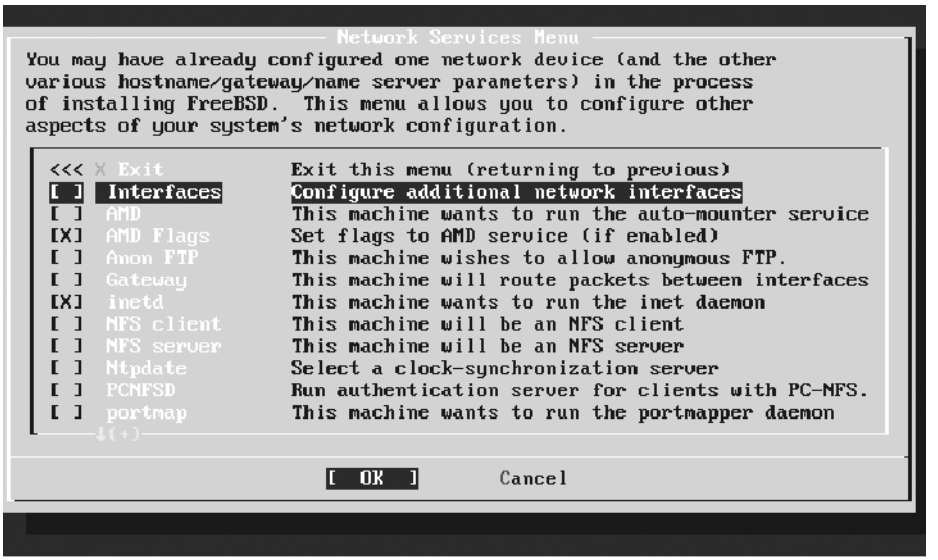

Рис. 1.17. *Меню Network Services*

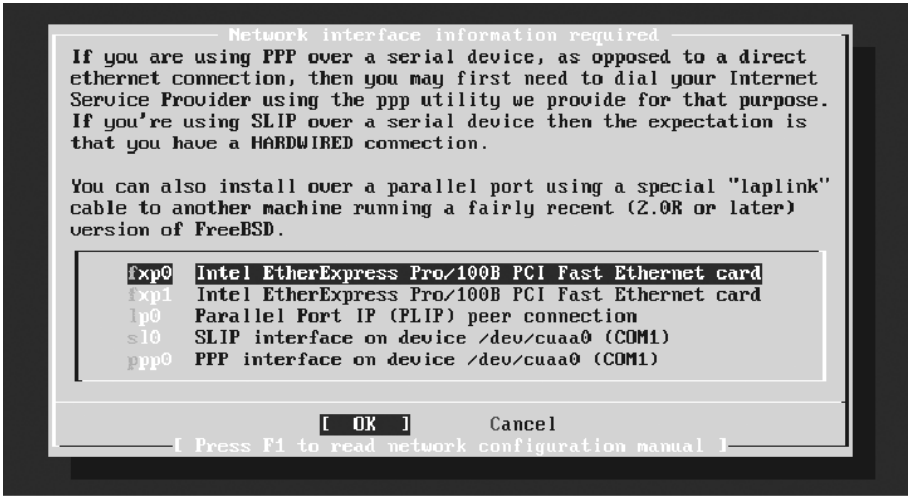

Рис. 1.18. *Меню информации о сетевых интерфейсах*

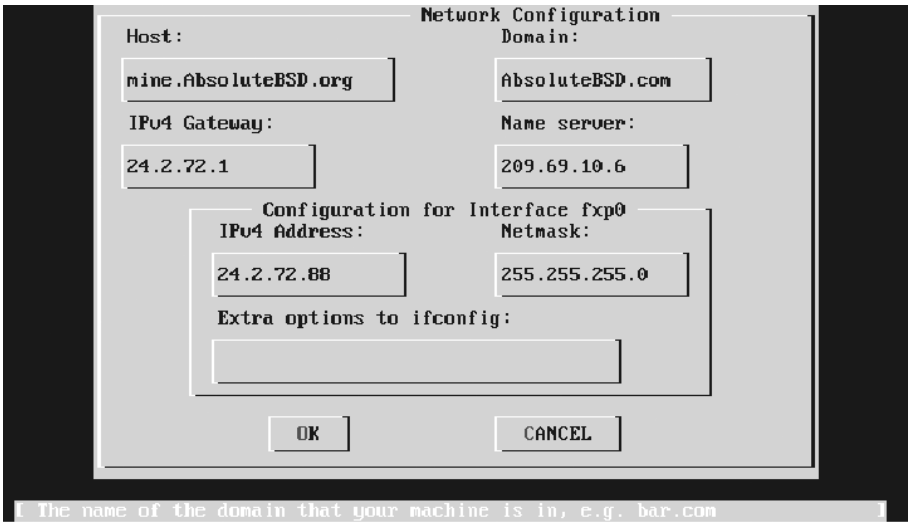

Рис. 1.19. *Меню Network Configuration*

Host name (Имя хоста) – уникальное имя компьютера. Это может быть что-либо подобное «Webserver» или «test». В имени должно быть только одно слово.

Domain name (Доменное имя) – домен, в который входит компьютер. Компьютер, используемый для примеров в этой книге, входит в домен AbsoluteBSD.com. Имя локального домена можно узнать у сетевого администратора.

В начале установки был дан совет узнать у сетевого администратора IP-адрес, сетевую маску, шлюз по умолчанию и IP-адрес сервера имен. Здесь следует ввести эту информацию.

Даже если возможно конфигурирование по DHCP, надо указать имя хоста и имя домена. В противном случае при загрузке система назовет себя «Amnesiac» (Утратившая память).

Xfree86

Опытный администратор UNIX, вероятно, обратит внимание на два пункта меню Configure Xfree86. XFree86 – это графический интерфейс пользователя (GUI), который рисует на мониторе приятные картинки (pretty pictures[1]). Примите совет – не выбирайте этот пункт сейчас. Бывают случаи, когда установка проходит неудачно из-за плохой конфигурации XFree86. Среду X всегда можно настроить после перезагрузки, применяя xf86cfg или другой инструмент. Так или иначе, среда X не приносит пользы на сервере. Вся ее работа сводится к потреблению системных ресурсов.

Среда X не будет обсуждаться в этой книге. Читатель, интересующийся этой темой, может обратиться к книге Билла Болла (Bill Ball) «The New Xfree86», выпущенной издательством Premier Press. Среда X – это не просто оконная система, подобная графическому интерфейсу Microsoft Windows; это целый протокол.

Программное обеспечение

Опытный администратор UNIX, вероятно, знает, какое программное обеспечение необходимо установить. Например, часто выбирают текстовый редактор Emacs. Необходимые программы можно установить, выбрав пункт Packages (Пакет программ) в меню настройки. В этом случае появится меню Package Selection (Выбор пакетов), представленное на рис. 1.20.

Читатель, уже знакомый с UNIX, вероятно, знает имена нескольких пакетов, которые надо установить. Один из популярных вариантов выбора – командная оболочка bash. Спуститесь в пункт shell, нажмите Enter, чтобы увидеть пакеты этой категории, выберите bash и нажмите пробел для выбора этого пункта. Далее нажмите Enter, чтобы вернуться в меню выбора пакетов.

Читателю, не знакомому с программным обеспечением UNIX, надо установить один пакет, который необходим для правильной работы с книгой. Выберите пункт Packages меню настройки, затем пункт net в меню выбора пакетов, а потом пункт cvsup (рис. 1.21). Инструмент cvsup будет применен в главе 6.

После выбора всех пакетов для установки надо вернуться в основное меню выбора пакетов. Нажмите Tab, чтобы переместить курсор с OK на Install, и нажмите Enter. Система начнет устанавливать пакеты.

---

[1] Согласно словарю жаргона Эрика Реймонда, такие картинки не всегда имеют что-либо общее с задачей, которую программа призвана решать. – *Примеч. перев.*

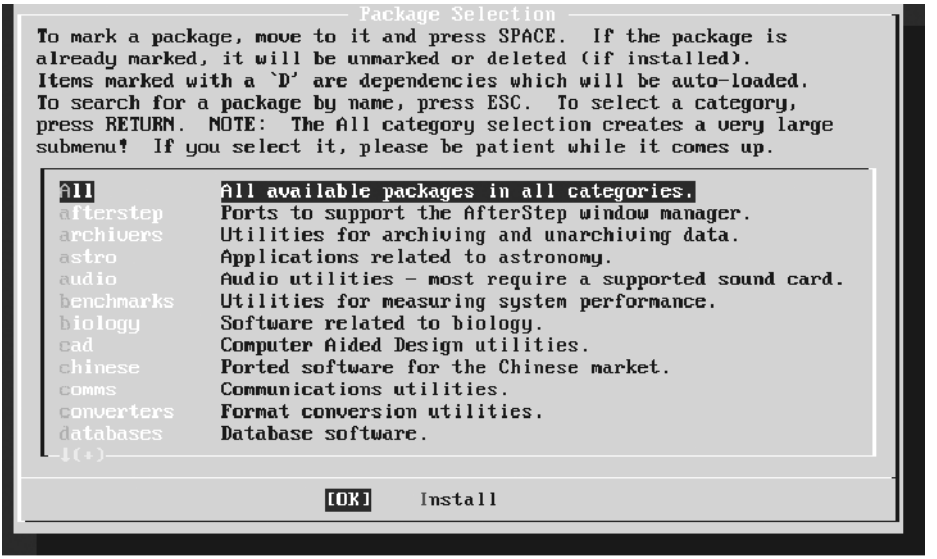

Рис. 1.20. Меню Package Selection

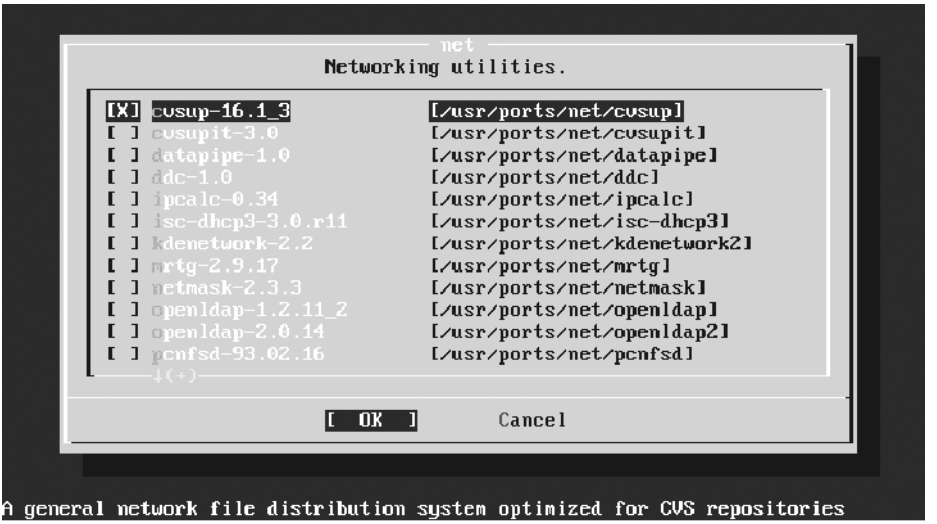

Рис. 1.21. Листинги отдельных пакетов

Перезапуск

Это последний шаг перед запуском системы. Извлеките из дисководов все компакт-диски и дискеты, выйдите из инсталлятора и перезагрузитесь. Теперь загрузилась полная система FreeBSD, сконфигурированная должным образом для работы в Интернете и для всех примеров этой книги.

При необходимости продолжить настройку системы программу sysinstall всегда можно запустить повторно:

```
# /stand/sysinstall
```

По ходу книги будут рассмотрены способы, позволяющие работать быстрее и эффективнее, не прибегая к sysinstall и обращаясь к файлам конфигурации напрямую. Однако программа *systinstall* оказывает существенную помощь в прохождении наиболее трудных этапов установки.

Замечание о редакторах

Нет, речь пойдет не о замечательных редакторах этой книги. Обсудим текстовые редакторы. Какой текстовый редактор UNIX считать «лучшим»? Этот вопрос является темой многолетних дебатов. Во FreeBSD есть *vi*, поскольку этот редактор распространяется на тех же правах, что и FreeBSD. Однако *vi* ужасает многих новичков; он ведет свою историю с эпохи возникновения UNIX. Это динозавр, точнее, велоцираптор (velociraptor[1]), небольшой, беспощадный и очень мощный – стоит только разобраться с его загадочным синтаксисом.

Если не подходит *vi*, то можно попробовать *ee* (Easy Editor). Это удобный редактор, который более подходит новичкам. По сравнению с *vi* его возможности более ограничены; устав от этих ограничений, можно перейти на *vi* или установить Emacs. (Я работаю с обоими, отдавая предпочтение Emacs.) Однако у *vi* есть бесспорное преимущество – он доступен на любых платформах UNIX, вот почему с ним надо уметь работать.

В большинстве программ будет применяться выбранный текстовый редактор, если в файл *.cshrc*, расположенный в домашнем каталоге, добавить следующую строку (заменив *vi* на выбранный редактор):

```
setenv  EDITOR vi
```

[1] Динозавр мелового периода. – *Примеч. перев.*

2

Как получить помощь

В книге такого объема, как эта, невозможно охватить все темы. В конце концов, система UNIX имеет 30-летнюю историю, BSD UNIX появилась более 20 лет назад, а FreeBSD известна 10 лет. Даже если запомнить весь материал книги, этого не будет достаточно для всех возможных ситуаций.

Проект FreeBSD поддерживает широкий круг информационных ресурсов, включая многочисленные почтовые рассылки FreeBSD, веб-сайт *www.FreeBSD.org*, Справочник (Handbook), FAQ (Frequently Asked Question, часто задаваемые вопросы), страницы руководства (man pages) и соответствующие веб-сайты пользователей. Море информации может подавлять и вызывать трудности в навигации. Однако прежде чем публиковать вопрос в почтовой рассылке, убедитесь, что необходимой информации нет в других источниках.

Почему не почтой с самого начала?

Почтовые рассылки FreeBSD – это замечательные ресурсы для получения технической поддержки. Многие специалисты, которые часто уделяют им внимание, обладают большим багажом знаний и могут быстро ответить на вопросы. Однако помните: размещение вопроса в почтовой рассылке FreeBSD означает, что один или несколько человек потратят свое время на ответ, вместо того чтобы посмотреть любимое телевизионное шоу, насладиться обедом в кругу семьи или отоспаться. Сложности возникают, когда на один и тот же вопрос эксперты отвечают 10, 50 и даже 100 и более раз. Они становятся раздражительными. Некоторые из них могут рассердиться не на шутку.

Плохо то, что эти люди затрачивают массу времени на подготовку ответов на эти вопросы – ответов, которые доступны в другом месте. Если очевидно, что запрашиваемых сведений нет в различных информационных источниках Проекта FreeBSD, спрашивающий наверняка получит вежливый полезный ответ. Однако отвечая на вопрос, который уже задавался несколько сотен раз, эксперт может не выдержать и показать свой буйный нрав.

Кроме того, следует помнить, что Проект FreeBSD поддерживает только FreeBSD. Почтовая рассылка FreeBSD – не то место, где следует просить по-

мощи при возникновении трудностей с другим программным обеспечением. Как правило, разработчики FreeBSD хорошо разбираются в различном программном обеспечении, однако это не означает, что они захотят помочь в настройке оконного менеджера WindowMaker X – тут надо побеспокоить специалистов, обслуживающих WindowMaker.

Выполняйте домашние задания и тогда появятся шансы, что ответ будет найден раньше, чем придет совет по почте.

Позиция FreeBSD

Для того чтобы успешно применять FreeBSD, надо выполнять *домашние задания*.

«Домашние задания? Что это значит, домашние задания? Назад в школу? Что вы хотите – огненных жертвоприношений и преклоненных колен?» Да, обратно в школу. В Проекте FreeBSD даже учителя по-прежнему ходят в школу. Заметим, однако, что цифровая передача огненных жертвоприношений затруднительна, да и неуместны они в современном мире.

Коммерческие операционные системы, такие как Windows 9x/NT, скрывают свою начинку. Все, к чему есть доступ, – это опции, представленные в GUI, и несколько инструментов командной строки, которые добавлены задним числом. Даже при желании *невозможно* понять, как работает тот или иной компонент. Когда происходит сбой, есть один выход – звонить производителю и униженно просить помощи. Хуже того, люди, оказывающие такую помощь за деньги, знают ненамного больше, чем пользователи, задающие вопросы.

С другой стороны, FreeBSD – полностью открытая система, которая позволяет глубоко изучить все подробности ее работы и предоставляет возможность ознакомиться с исходным кодом утилит и ядра. А специалисты из сообщества FreeBSD всегда готовы обучать. Они приветствуют людей, которые хотят учиться, и будут рады в этом помочь.

Существует другая часть сообщества FreeBSD, о которой надо помнить. Это специалисты, которые не заинтересованы в изучении работы FreeBSD. К их числу относится группа людей, которым нет дела до особенностей работы FreeBSD, если система работает хорошо. Например, многие поставщики услуг Интернета (Internet Service Providers, ISP) не заботятся о том, какая система обслуживает веб-страницы; им важно, чтобы веб-страницы обслуживались надежно. Программисты, разрабатывающие встроенные системы, зачастую так же не заинтересованы в изучении внутренней работы FreeBSD; большинство из них привлекает мощь этой системы и удобная лицензия. Речь не идет о том, что с этими группами специалистов что-либо не в порядке. Просто эти люди, вероятно, не заглядывают в почтовые рассылки FreeBSD, чтобы отвечать на вопросы пользователей.

Общее правило таково: люди помогают себе подобным. Пользователь FreeBSD должен перейти от рутины к чтению книги рецептов (cookbook) и приготовить свой собственный обед. Пользователя, желающего подробно изучить свою систему, ждут с распростертыми объятиями. Если же он просто хочет

знать, на какую кнопку нажать, ему следует обратиться к Справочнику или FAQ. У специалистов сообщества FreeBSD просто нет стимула помогать тем, кто не помогает себе сам или не способен следовать инструкциям. Если необходима более глубокая поддержка, которую не может предоставить сообщество, лучше всего заключить контракт для получения коммерческой поддержки. (Ее оказанием занимаются несколько компаний; дополнительную информацию можно найти на сайте FreeBSD.)

На самом деле специалистов, знающих FreeBSD досконально, можно пересчитать по пальцам. За неделю до написания этой главы я своими глазами видел одного из основателей Проекта FreeBSD, удивленного необычной работой программы. На душе стало легче – даже мастера учатся постоянно.

Итак, первая часть домашнего задания – изучить, какие ресурсы предоставляет Проект FreeBSD.

Страницы руководства

Страницы руководства (*man* pages; man – сокращение от «manual», руководство) – это первичная документация UNIX. Несмотря на то что эти страницы считаются непонятными, трудными для изучения, и говорят даже, что их невозможно читать, они вполне дружелюбны к пользователям – к особым пользователям. Когда создавались эти страницы, средний системный администратор писал программы на C. В результате эта документация была написана программистами для программистов. Пользователь, способный думать как программист, сочтет эти страницы совершенными. Я пробовал думать как программист, но реального успеха достиг лишь после двух дней непрерывного изучения документации (помогли сильное нервное возбуждение и кока-кола).

За последние несколько лет уровень квалификации, необходимый для системного администрирования, упал. Теперь системному администратору не обязательно быть программистом. Аналогичная вещь произошла со страницами руководства – читать их стало легче. По существу, они являются первой линией атаки при изучении работы системы. Пользователь, отправивший письмо в почтовую рассылку, не проверив эти страницы документации, наверняка получит краткий совет – заглянуть именно в них.

Руководство по FreeBSD

Руководство по FreeBSD состоит из девяти разделов. Каждая страница руководства представлена только в одном разделе. Эти разделы выглядят так:

1. General commands (Основные команды)
2. System calls and error numbers (Системные вызовы и коды ошибок)
3. The C libraries (Библиотеки C)
4. Devices and device drivers (Устройства и драйверы устройств)
5. File formats (Форматы файлов)

6. Game instructions (Инструкции к играм)
7. Miscellaneous information (Всякая всячина)
8. System maintenance commands (Команды обслуживания системы)
9. Kernel system interfaces (Системные интерфейсы ядра)

В страницах руководства номер раздела обычно указывается в круглых скобках после команды, например `reboot(8)`. Такая запись представляет как имя команды (`reboot`), так и страницу руководства (`8`). При поиске подробной информации о чем-либо надо обратиться к соответствующей странице документации. (Просмотреть страницу документации можно с помощью команды man(1).)

Страница документации есть почти по каждой теме. Например, для просмотра страницы руководства по редактору *vi* надо ввести следующую команду:

```
#man vi
```

В ответ система выдаст такую информацию:

```
VI(1)                                                    VI(1)


NAME (ИМЯ)
       ex, vi, view - text editors          (текстовые редакторы)

SYNOPSIS (СИНОПСИС)
       ex [-eFGRrSsv] [-c cmd] [-t tag] [-w size] [file ...]
       vi [-eFGlRrSv] [-c cmd] [-t tag] [-w size] [file ...]
       view [-eFGRrSv] [-c cmd] [-t tag] [-w size] [file ...]

LICENSE (ЛИЦЕНЗИЯ)
       The vi program is freely redistributable.  You are welcome to copy, modify
       and share it with others under the conditions listed in the LICENSE file.
       If any company (not individual!) finds vi sufficiently useful that you would
       have purchased it, or if any company wishes to redistribute it,
       contributions to the authors would be appreciated.
       (Перевод: Программа vi распространяется свободно, без ограничений. Допускается
       ее копирование, модификация и передача другим лицам согласно условиям, приве-
       денным в файле LICENSE. Если любая компания (не отдельное лицо!) сочтет про-
       грамму vi в достаточной мере полезной, она может купить ее. Если любая компания
       пожелает распространять vi, авторы будут признательны за полученные деньги.)

DESCRIPTION (ОПИСАНИЕ)
       Vi  is  a  screen oriented text editor.  Ex is a line-ori-
       (Vi – экранный текстовый редактор. Ex – строчный…)
:
```

Первый блок информации, представленный в верхней строке, сообщает заголовок страницы руководства и номер раздела. (Заголовок не обязательно соответствует набранной строке; например, при наборе `man ex` откроется страница руководства по *vi*.)

«Шапка» страницы руководства

В страницах руководства могут быть различные «шапки». Хотя в странице руководства может присутствовать название почти любого раздела, некоторые из них являются стандартными. (См. mdoc(7) для получения неполного списка и другой информации о стандартах страниц руководства.) Подобно авторам книг, авторы страниц руководства обычно располагают их содержимое в порядке, имеющем смысл для обсуждаемой программы. Тем не менее есть стандартные «шапки»:

- NAME сообщает различные имена команд. В приведенном примере vi, ex и view связаны друг с другом и потому им отведена одна страницу руководства. (Если немного копнуть, выяснится, что это одна и та же программа и ее поведение определяется способом запуска.) В разделе NAME также представлено краткое описание программы.
- SYNOPSIS приводит перечень возможных ключей командной строки и их аргументы. Зачастую этого раздела достаточно, чтобы освежить свою память и вспомнить ключ, с которым программа работала должным образом.
- DESCRIPTION содержит краткий обзор программы, которая описывается на странице руководства. Содержимое этого раздела зависит от темы, которую охватывает страница руководства, – к страницам о программах, файлах и базовых интерфейсах предъявляются несопоставимые требования.
- OPTIONS (ключи) описывает ключи программы, указываемые при запуске из командной строки, и их воздействие.

Дальнейшее обсуждение программы происходит в других основных разделах. Обычно это разделы BUGS (ошибки) и SEE ALSO (см. также).

В разделе BUGS описаны известные проблемы, связанные с программным кодом. Зачастую это позволяет избавиться от многих хлопот. Сколько вы бились, решая какую-нибудь задачу, только чтобы убедиться, что программа не работает так, как явствует из радостной, обнадеживающей документации? Цель раздела BUGS – описать известные ошибки и случаи странного поведения программы.[1]

SEE ALSO традиционно является последним разделом. Надо помнить, что система UNIX подобна языку и компоненты системы связаны в одно целое. Подобно Силе (Force) из «Звездных войн» (Star Wars) или клейкой ленте, ссылки из раздела SEE ALSO покажут, как компоненты соединяются в одно целое.

Иногда одно имя присутствует в нескольких разделах руководства. Так, в разделе 4 руководства есть страница для amd, посвященная драйверу AMD SCSI, а в разделе 8 есть страница для amd(8), описывающая программу для автоматического монтирования файловой системы. Чтобы получить страницу из конкретного раздела руководства, надо набрать его номер между командой man и названием страницы. Например, man 4 amd. (Это первое, на что следует

[1] Такая информация называется «честной» – в противовес «маркетинговой» информации, включаемой во многие другие программные продукты.

обратить внимание, если полученная страница содержит совершенно не то, что ожидалось.)

Чтобы увидеть все страницы, относящиеся к определенной теме, надо набрать `man -k`. В этом случае будет выполнен поиск по ключевому слову во всех заголовках страниц руководства. Для этой же цели также можно воспользоваться командой `apropos(1)` – это легче запомнить, если уже ясно, что означает «apropos».[1]

Для поиска слова в странице руководства следует набрать / и затем само слово. После этого на экране можно будет увидеть первое вхождение этого слова на странице. Последовательно нажимая клавишу `n`, можно переходить к последующим вхождениям этого слова.

Документация FreeBSD

Если документация FreeBSD установлена, ее можно найти в */usr/share/doc*. Здесь находятся несколько каталогов – для каждого языка, на который была переведена документация FreeBSD. Документация на английском языке расположена в каталоге *en*.

Документация FreeBSD разделена на *статьи* (*articles*) и *книги* (*books*). Различие между ними весьма условно. Как правило, книги больше статей и охватывают более широкие темы, а статьи короче и описывают один предмет. Две книги, наиболее интересные новым пользователям, – это Handbook (Справочник) и FAQ. Обе книги доступны в режиме онлайн по адресу *http://www.FreeBSD.org*. Справочник – это непрерывно изменяющийся путеводитель Проекта FreeBSD. Он описывает методику выполнения основных задач в системе и является хорошим подспорьем в начале работы. FAQ (Frequently Asked Question, часто задаваемые вопросы), подобно Справочнику, разделен на темы, но кроме этого содержит ответы на вопросы из почтовых рассылок. Часть информации из FAQ продублирована в Справочнике, но далеко не вся.

Если задача понятна и сформулирован конкретный вопрос по поводу обнаруженной ошибки, следует обратиться к FAQ. Если вы не понимаете, что надо делать, обратитесь к Справочнику. Если Справочник не помогает, поищите статью о том, чего предполагается достичь. (Поиск по всему комплекту документации FreeBSD можно провести по адресу *http://www.FreeBSD.org/search/*. Если и это не поможет, обратитесь к архивам почтовых рассылок на той же странице.)

Архивы почтовых рассылок

Если возникшая проблема не вписывает новую страницу в историю FreeBSD, значит, кто-то с ней уже сталкивался и, вероятно, рассказал об этом в почтовой рассылке. Как-никак, появление архивов восходит к 1994 году и

[1] Apropos (англ.) – здесь: относительно, по поводу; что касается. – *Примеч. перев.*

они содержат около миллиона сообщений. Конечно, найти необходимую информацию среди миллиона сообщений – задача непростая.

Оказавшись в тупике, разместите сообщение об ошибке в окне поиска по архиву почтовых рассылок, расположенному по адресу *http://www.FreeBSD.org/search/*. Удалите служебные слова, такие как «and» (и), «or» (или), «but» (но) и тому подобные, и нажмите кнопку поиска. (Эта страница предусмотрена для поиска в почтовой рассылке FreeBSD-questions (вопросы по FreeBSD).) Если результат неудовлетворительный, в окно поиска надо ввести другое сообщение или фразу, которую компьютер выдает при поиске ошибки. Исчерпывающий ответ вполне можно получить со второй или третьей попытки.

Если в архивах с вопросами по FreeBSD не найдено ничего полезного, можно попробовать другие рассылки, относящиеся к теме. Вот некоторые достойные внимания: FreeBSD-hackers, FreeBSD-stable и FreeBSD-current.[1] Если возникшие трудности связаны с конкретной подсистемой, надо обратиться к рассылке, посвященной этой теме.

Примечание

Поля ввода для поиска по документации и почтовым рассылкам находятся на одной странице! Не один раз я ошибался, пытаясь искать по документации, хотя собирался искать в архивах почтовых рассылок.

Другие веб-сайты

Если ответ на вопрос не найден, можно обратиться к другим веб-сайтам.

FreeBSD Diary (*http://www.FreeBSDdiary.org*)

На этом сайте детально изложен опыт пользователей FreeBSD. Статьи содержат подробное описание действий, необходимых для надлежащей работы системы.

Google (*http://www.google.com*)

На этом сайте собраны архивы телеконференций Usenet по многим темам, включая FreeBSD. Попробуйте мощный поиск в архивах *mailing.FreeBSD.** или *comp.unix.bsd.FreeBSD.**. Здесь можно найти материалы как почтовых рассылок, так и телеконференций. (Кроме того, на Google по адресу *http://www.google.com/bsd* расположена поисковая машина, специфичная для FreeBSD.)

Daemonnews (*http://www.daemonnews.org*)

Популярный сайт новостей BSD. Ежемесячные выпуски содержат множество статей по различным темам, относящимся к BSD.

[1] Русскоязычный читатель может найти ответ на свой вопрос или решение проблемы еще и в архивах веб-сайта *www.opennet.ru*. – *Примеч. науч. ред.*

FreeBSD 'zine (*http://www.FreeBSDzine.org*)

Сайт статей по FreeBSD, выходящих дважды в месяц. Много полезного материала.

Defcon1 (*http://www.defcon1.org*)

Другой сайт со статьями по FreeBSD.

BSD Today (*http://www.bsdtoday.com*)

На сайте *Internet.com* можно найти статьи по BSD и ссылки к новостям.

O'Reilly Network BSD Developer Center (*http://www.onlamp.com/bsd*)

Здесь размещено множество статей по BSD, включая колонку вашего покорного слуги «Big Scary Daemons» (Большие ужасные демоны).

Использование ресурсов FreeBSD, связанных с принятием решений

Теперь возьмем простую задачу и воспользуемся ресурсами FreeBSD для ее решения. Для нахождения ответа применим несколько методов. Вот типичное сообщение, посланное по адресу *FreeBSD-questions@FreeBSD.org*:

«Я только что установил FreeBSD и моя сеть не работает. После выполнения команды `ping` на консоли появляется сообщение `ed0: device timeout`. В чем тут дело?»

Поиск в Справочнике/FAQ

Беглый просмотр Справочника ничего не дал. Однако в FAQ, в разделе Troubleshooting (Устранение неполадок) есть статья:

```
I keep seeing messages like ed1: device timeout
(Я постоянно вижу сообщения типа ed1: device timeout)
```

Достаточно близко. Прочитайте статью и попробуйте применить предложенное решение.

Поиск в страницах документации

Далее будет объяснено, что цифры после имени устройства (device name) просто указывают на конкретное устройство. `ed0` — это устройство `ed` с номером 0. Поэтому надо набрать `man ed`. На экране появится следующая информация:

```
ED(1)                  FreeBSD General Commands Manual                  ED(1)
                      (Руководство по основным командам FreeBSD)

NAME (ИМЯ)
     ed, red - text editor

SYNOPSIS (СИНОПСИС)
     ed [-] [-sx] [-p string] [file]
```

```
DESCRIPTION (ОПИСАНИЕ)
     Ed is a line-oriented text editor.  It is used to create, display, modify
     and otherwise manipulate text files.
     (Перевод: Ed - это строчный[1] текстовый редактор. Он применяется для создания,
     отображения, модификации текстовых файлов и других операций с ними.)

     If invoked with a file argument, then a copy of file is read into the editor's
     buffer.  Changes are made to this copy and not directly to file itself.
     Upon quitting ed, any changes not explicitly saved  with a w command are lost.
     (Перевод: Если при вызове ed в качестве аргумента указать имя файла, копия файла
     окажется в буфере редактора. Изменения будут выполняться в этой копии,
     а не в самом файле. После выхода из ed все изменения, явно не сохраненные
     с помощью команды w, будут потеряны.)

     Editing is done in two distinct modes: command and input.  When first invoked,
     ed is in command mode.  In this mode commands are read from the standard input
     and executed to manipulate the contents of the editor buffer. A typical command
     might look like:
     (Перевод: Редактирование выполняется в двух отдельных режимах: командном режиме
     и режиме ввода. Первоначально ed находится в командном режиме. В этом режиме
     команды, предназначенные для манипулирования содержимым буфера редактора,
     считываются со стандартного устройства ввода и выполняются. Типичная команда
     может выглядеть так:)

     ,s/old/new/g
:
```

В чем дело? Здесь явно что-то не так. Для каждого драйвера устройства есть страница руководства. Запустите `man -k ed`, чтобы увидеть полный список всех страниц руководства, относящихся к ed. Появится полный список функций в слегка искаженном алфавитном порядке (заглавные буквы идут перед строчными). Прокрутите текст вниз до «e». Вот что будет на экране:

```
...
ed(1), -(1)              - ed text editor (текстовый редактор ed)
ed(4)                    - high performance ethernet device driver (драйвер
                           высокопроизводительного устройства ethernet)
...
```

Ага! Есть два различных ed, в разных разделах. Для каждого есть своя страница руководства. Наберите `man 4 ed`, и на экране появится необходимая информация:

```
ED(4)                  FreeBSD Kernel Interfaces Manual                  ED(4)
                      (Руководство по базовым интерфейсам FreeBSD)

NAME (ИМЯ)
     ed - ethernet device driver     (ed - драйвер устройств ethernet)
```

[1] Мы привыкли к полноэкранным текстовым редакторам, позволяющим редактировать весь текст в целом и видеть его при этом на экране. Строчный редактор позволяет просматривать и редактировать только одну строку в каждый момент времени. – *Примеч. науч. ред.*

```
SYNOPSIS (СИНОПСИС)
     device ed          (устройство ed)

DESCRIPTION (ОПИСАНИЕ)
     The ed driver provides support for 8 and 16bit ethernet cards that are based
     on the National Semiconductor DS8390 and similar NICs manufactured by other
     companies.
     (Перевод: Драйвер ed поддерживает 8- и 16-разрядные карты ethernet на базе чипа
      DS8390 компании National Semiconductor и подобные сетевые адаптеры, производи-
      мые другими компаниями.)

     It supports all 80x3 series ethernet cards manufactured by Western Digital and
     SMC, the SMC Ultra, the 3Com 3c503, the Novell NE1000/NE2000 and compatible
     cards, and the HP PC Lan+.  ISA, PCI and PC Card devices are supported.
     (Перевод: Он поддерживает все карты ethernet серии 80x3, производимые компаниями
     Western Digital и SMC, SMC Ultra, 3c503 компании 3Com, Novell NE1000/NE2000
     и совместимые с ними карты, а также PC Lan+ компании HP. Поддерживаются
     устройств ISA, PCI и PC Card.)

     The ed driver uses a unique multi-buffering mechanism to achieve high transmit
     performance.  When using 16bit ISA cards, as high as 97% of the theoretical
     maximum performance of the IEEE 802.3 CSMA ethernet is possible.
     (Перевод: Для высокопроизводительной передачи данных драйвер ed применяет
     уникальный механизм, основанный на использовании нескольких буферов. В случае
     16-разрядных карт ISA производительность может составить 97% от максимальной
     теоретической производительности, предусмотренной стандартом IEEE 802.3 CSMA
     для сетей ethernet.)
:
```

Именно то, что нужно. Обратившись к сообщению об ошибке, можно догадаться, что timeout – хорошее ключевое слово. Наберите `/timeout` и нажмите Enter.

```
 ed%d: device timeout Indicates that an expected transmitter interrupt didn't occur.
       Usually caused by an interrupt conflict with another card on the ISA bus.
(ed%d: тайм-аут в работе устройства свидетельствует, что ожидаемого прерывания
       от передатчика не произошло. Обычно это вызвано конфликтом по прерываниям
       с другой картой на шине ISA.)
```

Вот так! Здесь представлено краткое описание проблемы и вызвавшая ее причина (конфликт по прерываниям). Старая добрая проблема с IRQ.

Поиск в архивах почтовых рассылок

Поиск в архивах почтовых рассылок выражения «ed0: device timeout» выдаст довольно много сообщений. В тот день, когда я искал, первое же сообщение содержало решение.

Использование ответа

Любой ответ, полученный на вопрос о «ed0 timeout», подразумевает, что спрашивающий знает, что такое IRQ и как настраивать прерывания. Эти знания соответствуют типичному уровню квалификации, необходимому для

решения основных задач. Если полученный ответ неясен, эту тему следует изучить и понять.

Несмотря на то что опытный разработчик или системный администратор, вероятно, не загорятся желанием объяснять, что такое IRQ, они охотно сообщат адрес веб-страницы, где рассматривается эта тема.[1]

Письмо о помощи

Если архивы, FAQ, Справочник, пособия и другие источники не помогли, надо обратиться за помощью. Обязательно включите в письмо всю информацию, имеющуюся в распоряжении (это мы обсудим чуть ниже). И той, которую можно было бы включить в письмо, довольно много. Этого можно не делать, но тогда случится одно из двух:

- Вопрос будет проигнорирован.
- Придет масса сообщений, требующих эту информацию собрать.

Если же помощь действительно необходима, то в сообщение надо включить следующее:

- Полное описание проблемы. Сообщение вроде «Как заставить модем работать?» вызовет массу вопросов. Что требуется от модема? Модем какого типа? Каковы симптомы?
- Лучше всего начать письмо так: «Мой модем не устанавливает соединение с провайдером. Модем BleahCorp v.90 model 6789. Версия ОС – 4.6-stable. Материнская плата с двумя процессорами Athlon. В файле */var/log/ppp.log* сообщений об ошибках нет». Таким образом можно избежать потока писем со встречными вопросами. И хотя мне еще не приходилось видеть кого-либо, кто был бы недоволен слишком большим объемом информации в почтовой рассылке FreeBSD-questions, следует поступать именно так, а не иначе.
- Вывод команды `uname -a`. В нем сообщается версия операционной системы и платформа.
- В случае CVSup и «make world» надо указать дату последнего CVSup.

  Дата CVSup (в секундах) с 1 января 1970 года – это третье поле в первой строке файла протокола. (Конечно, если исходный код был обновлен без «make world», это может быть не так.) Например, на моей системе CVSup записывает данные в */usr/sup*. Первая строка в файле */usr/src/src-all/check-outs.cvs:RELENG_3/* выглядит так:

  ```
  F 5 939160270
  ```

  Запуск `date -r 939160270` выдаст следующее:

  ```
  Tue Oct  5 21:51:10 GMT 1999
  ```

---

[1] Не стоит на это рассчитывать. На их месте я бы посоветовал зайти на *google.com* и там поискать, что такое IRQ. – *Примеч. науч. ред.*

- Любые сообщения об ошибках – чем полнее, тем лучше. Необходимо включить все сообщения из файлов протоколов, особенно из */var/log/messages*.

Наконец, вот несколько подсказок, следование которым поможет получить наилучшие результаты при обращении к почтовой рассылке FreeBSD:

- Будьте вежливы. Помните, что рассылка поддерживается добровольцами, которые отвечают на письма лишь из добрых побуждений. Перед тем как нажать «Send» (Отправить), спросите себя: «А из-за этого письма я не опоздаю на свидание с двойняшками?».[1]
- Письмо должно содержать только текст (plain text). Многие разработчики FreeBSD читают письма в среде командной строки (command-line environment) и чтение файлов HTML их раздражает. (Чтобы убедиться в этом, установите */usr/ports/mail/mutt* и попробуйте почитать в нем письма в формате HTML.)
- Точно укажите, что было предпринято для решения проблемы и ответа на этот вопрос. Покажите, что домашнее задание выполнено.
- Пишите по теме. Если есть трудности с XFree86, следует зайти на сайт XFree86. Если не работает менеджер окон, то спрашивать надо людей, которые его поддерживают. Просить у специалистов по FreeBSD помощи в конфигурировании Java Application Server – все равно, что жаловаться поставщикам аппаратного обеспечения на обед в экспресс-закусочной. Может быть, у них и есть лишний кетчуп, но это не их проблема. С другой стороны, если система FreeBSD запускает sendmail при каждой загрузке и это надо отменить, следует сначала изучить онлайновые источники и уже затем просить помощи.
- Пошлите письмо по адресу *FreeBSD-questions@FreeBSD.org*. Да, есть другие почтовые рассылки FreeBSD, которые, возможно, посвящены как раз тому, с чем возникли трудности. Однако вопросы новичков почти всегда лучше всего подходят для FreeBSD-questions. В течение многих лет я отслеживал рассылки FreeBSD-stable, -current и -hackers и должен заметить, что новички задают вопросы, на которые лучше ответят в FreeBSD-questions. Обычно задающих такие вопросы отсылают к -questions.

  Если сообщение с вопросом об установке коммутируемого соединения отправить в FreeBSD-hackers, хакеры будут раздражены. Ответ может быть получен, но друзей у вас не прибавится. Наоборот, специалисты, поддерживающие рассылку FreeBSD-questions, добровольно отвечают на вопросы. Они рады им. Многие из них разрабатывают FreeBSD, а некоторые даже являются ключевыми игроками. Одни добровольцы – высококвалифицированные специалисты, другие – новички, уже имевшие дело с подобными проблемами.

---

[1] Найдется довольно много разработчиков, готовых принять вместо проявлений вежливости телефоны упомянутых двойняшек. Но это не гарантируется и принято только у очень продвинутых специалистов.

Если они не смогут помочь, то наверняка порекомендуют другую почтовую рассылку. Намного лучше обратиться в рассылку -hackers и сказать: «В -questions посоветовали обратиться к вам», чем сразу писать в -hackers. Сообщество FreeBSD уважает тех, кто уважает сообщество.

- Следуйте указаниям. Если в ответе на вопрос запрашивается дополнительная информация, ее следует предоставить. Если неясно, как ее получить, это другая проблема. Обратитесь к началу этой главы и попробуйте найти решение. Если запрашивается дамп в режиме отладки, просмотрите главу 16 и настройте систему для получения такого дампа. Другими словами, если станет ясно, что задающий вопрос не предоставляет запрашиваемой информации, он вообще не дождется ответа.
- Наконец, ответ на вопрос: «Где можно почитать про это?» более вероятен, чем ответ на вопрос: «Что мне делать?». Неважно, относится ли вопрос к непонятным системным функциям или к простому поиску неполадок; готовность потрудиться для получения ответа необходима при работе с FreeBSD. Зато когда ответ найден, система становится гораздо более понятной, чем была до этого.

Итак, теперь ясно, куда обратиться в случае, если материала книги будет недостаточно, и мы поговорим о том, как защитить себя от собственных ошибок.

3

Прочтите это раньше, чем что-нибудь испортите! (Резервное копирование и восстановление)

Сбои в работе компьютеров могут происходить на многих уровнях. Аппаратное обеспечение и програмы, пользователи и системные администраторы – все это может вызывать нарушения работы системы. Всегда надо быть готовым к худшему; в случае с компьютерами это означает готовность проводить резервное копирование и восстановление данных с жестких дисков.

Поскольку FreeBSD – непрерывно развивающаяся система, время от времени ее неизбежно придется обновлять и накладывать на нее «заплатки». При этом всякий раз существует опасность нанести вред операционной среде. Если это случится, возникнет необходимость восстановить систему или скомпоновать ее заново. (Достаточно вспомнить, как часто приходилось накладывать заплатки на серверы с системой от Microsoft и сталкиваться впоследствии со странным поведением системы.) На любом компьютере даже минимальные изменения конфигурации потенциально способны повредить данные.

Хуже того, читатель этой книги, вероятно, как раз изучает, как настраивать систему FreeBSD и, возможно, не готов к аварии. Новичку необходимо протестировать множество вариантов конфигурации и проанализировать «историю» изменений в конфигурации системы. Если выясняется, что какая-то непонятная, но важная системная функция месяцами не действует должным образом, необходимо проанализировать проведенные изменения, чтобы вернуться назад и восстановить работоспособность. Помнит ли читатель, как выглядел тот или иной файл неделю или месяц назад? На самом деле интенсивные эксперименты могут полностью нарушить работу системы, поэтому необходим способ восстановления важных данных.

Эта глава начинается с рассмотрения общего подхода – резервного копирования всего, что есть в компьютере. Однако такой подход недостаточно хорош для сохранения отдельных файлов, поэтому далее будут рассмотрены другие, более удобные способы. Если файл изменяется три раза в день, а резервное копирование происходит раз в неделю, то можно потерять важные данные. В заключение будут рассмотрены меры по восстановлению и переком-

поновке системы в случае частичных неполадок – с применением однопользовательского режима и загрузочной дискеты для ремонтных работ (fixit).

Резервное копирование системы

Резервное копирование системы необходимо, лишь если данные важны. Вопрос, который надо себе задать, таков: «Сколько будет стоить восстановление данных?» Например, недорогая система резервного копирования на ленте с интерфейсом SCSI стоит несколько сотен долларов. Системы IDE дешевле, чем SCSI, однако они работают медленнее, позволяют сохранять меньше данных и хуже поддерживаются.

Выбирая вариант резервного копирования, читатель должен задать себе вопрос, сколько стоит его время и сколько времени может отнять восстановление системы с выбранного носителя. Если самые важные данные на компьютере – это файл с закладками веб-броузера, возможно, нет смысла приобретать систему резервного копирования. Однако если сервер – это остов компании или его часть, об инвестировании следует позаботиться очень серьезно.

Для операций по полному сохранению и восстановлению необходим накопитель на магнитной ленте. Кроме этого, данные можно сохранять в файлах, проводя резервное копирование по сети, или на съемных носителях, таких как компакт-диск или дискета. Однако, поскольку в книге обсуждаются сетевые серверы, внимание здесь будет сосредоточено на серийных решениях.

Накопители на лентах

FreeBSD поддерживает накопители на магнитных лентах с интерфейсами SCSI и IDE. По сравнению с устройствами IDE устройства SCSI быстрее и надежнее, однако устройства IDE дешевле. В большинстве случаев подойдет любое из них.

После физической установки накопителя на ленте необходимо убедиться, что FreeBSD распознала это устройство. Самый простой путь – проверить файл */var/run/dmesg.boot*, в котором отображены сообщения об ошибках при загрузке системы и показано все аппаратное обеспечение, присутствующее в системе. Это очень большой файл, поэтому здесь он не приводится. Обязательно изучите *dmesg* как следует, поскольку при возникновении неполадок в аппаратных средствах обращаться надо к нему.

Исследуя этот файл, можно увидеть IDE-накопители на ленте, отображаемые как устройства «ast», и накопители SCSI, представленные как «sa». Поищите в этом файле установленный накопитель на ленте; если он есть, то ядро системы, вероятно, сконфигурировано должным образом. (Ядро GENERIC FreeBSD поддерживает большинство накопителей на ленте.)

Как читать Dmesg.boot

В приведенном далее примере boot-файла первое поле в каждой строке содержит имя устройства. Запись в этом конкретном примере представляет

накопитель на ленте DDS3, который считается достаточно медленным по современным стандартам, но по-прежнему отвечает предъявляемым требованиям. Если в системе есть несколько накопителей на ленте, у них будут последовательные имена, например sa0, sa1, sa2 и т. д. Зная имя устройства, можно получить доступ к накопителю на ленте. Устройства SCSI представлены буквами «sa» и замыкающим номером; накопители IDE показаны как «astX» вместо «saX». Если есть несколько идентичных устройств, их следует промаркировать. Если неясно, где накопитель на ленте подсоединен к физической шине, или неизвестен идентификатор устройства SCSI, каждый накопитель можно распознать методом проб и ошибок. (Каждый раз, подготавливая новый сервер копирования, я собираюсь маркировать накопители на лентах, нанося идентификаторы устройств SCSI в соответствии с порядком их подсоединения к машине. Но в конце концов помещаю ленту в каждый накопитель и пытаюсь получить к нему доступ под различными номерами,[1] выясняя соответствие опытным путем. Годятся оба подхода.)

Каждая строка, относящаяся к устройству, содержит информацию об этом устройстве. Например, для накопителя на ленте, представленного в следующем примере, есть три описательных строки. Они сообщают о том, как накопитель подсоединен к системному интерфейсу SCSI, название модели и максимальную скорость.

```
sa0 at ahc0 bus 0 target 6 lun 0
sa0: <ARCHIVE Python 04106-XXX 7350> Removable Sequential Access SCSI-2 device
sa0: 10.000MB/s transfers (10.000MHz, offset 15)
```

Каждое устройство в системе, от микросхемы часов на плате и шин PCI до звуковых карт, будет иметь похожие записи в *dmesg.boot*.

Файл *dmesg.boot* – неоценимый источник информации об установленных аппаратных средствах. Загляните в этот файл, – кто бы мог подумать, что в этом маленьком бежевом корпусе скрыто столько компонентов.

Если накопитель на ленте не обнаружен в файле */var/run/dmesg.boot*, то надо ознакомиться с замечаниями по конкретной версии FreeBSD, чтобы выяснить, поддерживается ли этот накопитель. Если оказалось, что такое устройство поддерживается, но его нет в файле *dmesg.boot*, следует обратиться за помощью (см. главу 2). Вероятно, будет получен совет пересобрать ядро (см. главу 4).

Управление накопителем на ленте

Накопители на магнитных лентах применяются много лет и способ управления ими, реализованный во FreeBSD, отражает их историю. Подобно многим другим старомодным устройствам UNIX, способ доступа к накопителю на ленте определяет его поведение. Перед применением такого накопителя необходимо научиться им управлять. Основной механизм управления нако-

[1] То есть указывая разные номера устройств типа rmt0, rmt1 и т. д. – *Примеч. науч. ред.*

пителем – применение файла устройства, который используется для доступа к нему.

Файлы устройств

Файлы устройств, расположенные в каталоге */dev*, – это файлы, привязанные к физическим устройствам компьютера. Файлы устройств могут выступать в качестве аргументов команд управления аппаратными средствами, однако такие команды нельзя выполнять произвольно; что-нибудь подобное `cat /dev/console` может ни к чему не привести, а может и повредить аппаратным средствам или данным. В большинстве случаев имена файлов устройств совпадают с названиями устройств в *dmesg.boot* или похожи на них.

Для каждого типа накопителей на ленте есть несколько файлов устройств, но при наличии обычного SCSI-накопителя интерес представляют только три файла: */dev/esa0*, */dev/nsa0* и */dev/sa0*. Точно так же, для IDE-накопителя важны файлы */dev/east0*, */dev/nast0* и */dev/ast0*.

Если в команде указывается файл устройства, который соответствует имени устройства, лента будет автоматически перемотана после завершения операции. Возьмем устройство SCSI sa0; при запуске команды, в которой в качестве файла устройства указан */dev/sa0*, лента будет автоматически перемотана по завершении команды. В зависимости от операционной системы, к которой привык читатель, это может либо соответствовать, либо не соответствовать его ожиданиям. В разных версиях UNIX, в которых для управления накопителями на лентах применяется различное программное обеспечение, работа с лентами организована по-разному.

Совет

Накопители на лентах – это устройства с последовательным доступом; данные на ленте хранятся линейно. Для получения доступа к определенным данным на ленте необходимо перематывать ленту вперед и назад. Делать это или не делать – вопрос важный.

Для того чтобы лента извлекалась автоматически после завершения операции, следует указать файл устройств, имя которого начинается с «e». Так, если операция подразумевает резервное копирование всей системы, для автоматического извлечения ленты по окончании сохранения следует задействовать устройство */dev/esa0*. (Некоторые старые накопители на лентах могут не поддерживать автоматическое извлечение ленты; в этом случае необходимо нажать клавишу для приведения в действие рычага, который извлечет ленту из накопителя. Самый простой способ, позволяющий выяснить, возможно ли автоматическое извлечение ленты, состоит в том, чтобы попробовать.)

Бывает так, что после завершения операции ленту перематывать не надо,– например, когда предполагается выполнить резервное копирование данных с другой машины и разместить их на ленте. В этом случае следует задействовать файл устройства, имя которого начинается с «n». Скажем, если в команде указан */dev/nsa0*, лента не будет перемотана.

Использование переменной TAPE

Во многих программах подразумевается, что накопитель на ленте – это */dev/sa0*, однако такое предположение не всегда верно. Даже если в системе установлен только один накопитель SCSI, то возможно, что автоматическая перемотка ленты не нужна (*dev/nsa0*) или надо извлечь ленту по завершении резервного копирования (*/dev/esa0*). Либо в системе есть накопитель IDE, для которого используется совершенно другой файл устройства.

При выборе файла устройства многие программы ориентируются на переменную окружения $TAPE, значение которой можно всегда изменить из командной строки. Большинство программ для резервного копирования по умолчанию будут использовать файл устройств, указанный в $TAPE.

Значение переменной $TAPE можно установить следующей командой:

```
# setenv TAPE /dev/sa0
```

Примечание

Переменную $TAPE распознают не все программы, но обычно есть смысл ее установить.

Команда mt

Если известен файл устройства, который надо задействовать для обращения к накопителю на ленте, то основные операции (такие как перемотка, натяжка, очистка и т. д.) можно выполнять с помощью команды mt(1). Чаще всего команда mt применяется для выяснения состояния накопителя на ленте:

```
# mt status
Mode      Density              Blocksize      bpi      Compression
Current:  0x25:DDS-3           variable       97000    DCLZ
---------available modes---------
0:        0x25:DDS-3           variable       97000    DCLZ
1:        0x25:DDS-3           variable       97000    DCLZ
2:        0x25:DDS-3           variable       97000    DCLZ
3:        0x25:DDS-3           variable       97000    DCLZ
---------------------------------
Current Driver State: at rest.
---------------------------------
File Number: 0  Record Number: 0        Residual Count 0
#
```

Большая часть информации, представленной в выводе mt, особого значения не имеет. Однако если эти данные необходимо изучить построчно, следует обратиться к странице документации mt(1), хорошо описывающей все особенности команды. Прежде всего в выводе mt обратите внимание, что эта команда «находит» накопитель. То есть для применения накопителя система настроена должным образом. (В данном примере команда mt обнаружила, что накопитель на ленте имеет тип DDS-3.) Различные «режимы», представленные в выводе mt, означают варианты работы накопителя. Поле Current Driver State сообщает, чем занимается устройство в данный момент.

При первом запуске `mt status` можно увидеть примерно следующее:

```
# mt status
mt: /dev/nsa0: Device not configured
```

Это означает, что в системе нет накопителя на ленте, соответствующего файлу устройства, который указан в переменной `$TAPE`. С файлами устройств можно поэкспериментировать, если посредством ключа `-f` указать файл устройства (например, `mt -f /dev/nsa1 status`). Однако необходимую информацию можно получить из *dmesg.boot*.

Другие полезные варианты запуска команды mt: `mt rewind`, `mt offline` и `mt retension`. Очевидно, что `mt rewind` перематывает ленту, `mt offline` извлекает ее, а `mt retension` натягивает ленту, прогоняя ее по всей длине – как вперед, так и назад. (Натяжка зачастую необходима, поскольку лента растягивается при ее первом использовании; натяжка заранее растягивает ленту перед тем, как на нее будут записаны данные.)

Примечание

Не все накопители на ленте поддерживают все функции mt. Многие старые устройства весьма темпераментны. При возникновении трудностей с накопителем поищите в архиве почтовой рассылки FreeBSD-questions сообщения от специалистов, работавших с таким устройством. Возможно, ответ будет найден именно там.

Итак, имя накопителя и способы управления этим устройством известны, и все готово для резервного копирования системы.

Программы для создания резервных копий

Два популярных пакета для резервирования системы – это tar(1) и dump(8). Разумеется, кроме dump есть и другие инструменты, которые можно применять во FreeBSD, например Amanda, pax и cpio. Подобные инструменты хорошо подходят для определенных сред, но они не везде так полезны, как tar и dump. Впрочем, если освоить dump и tar, то любые другие программы покажутся сравнительно более легкими.

Программа tar предназначена для работы с файлами. Данные из резервных копий, созданных с помощью tar, можно восстановить почти на всех операционных системах. Программа dump работает с дисковыми разделами и файловыми системами. Информацию из резервных копий, полученных командой dump, можно восстановить исключительно на той же операционной системе, на которой были созданы копии. Для резервного копирования всех файлов следует применять dump. Если необходимо сохранить небольшой объем данных либо их потребуется восстановить в совершенно другой среде, следует предпочесть tar.

tar

Пакет tar, сокращение от «tape archiver» (ленточный архиватор), может создавать резервные копии любой информации – от одного файла до всей систе-

мы. tar работает с файлами и каталогами и понятия не имеет о базовой файловой системе (это его преимущество и недостаток). tar – это общий стандарт, признанный почти всеми производителями UNIX и другого программного обеспечения. Программу tar можно запускать в средах Windows, Linux, UNIX, BSD и почти во всех других операционных системах. Файлы из архива tar можно извлекать даже на машине Windows!

tar может сохранять файлы на ленте или в специальном файле. Такой файл называется *tarball*. Поскольку tar работает с файлами, извлечь и восстановить файл из tarball очень легко.

FreeBSD применяет инструмент GNU tar, основанный на старой свободно распространяемой программе tar. При работе с GNU tar могут возникнуть трудности с необычными файлами (например, с пустыми). Если какая-либо программа создает необычные файлы, то в ее документации обычно указано, что для резервного копирования следует применять не tar, а что-нибудь другое. А если файловая система повреждена, то одному богу известно, что сохранит tar.

Один из недостатков tar – молчаливость; например, он «успешно» восстановит файлы, которые были повреждены при первоначальном копировании (хотя это редко случается на практике).

Режимы tar

tar имеет несколько общих режимов, которые задаются ключами при вызове программы из командной строки. (См. tar(1), где приведено описание всех режимов; здесь будут рассмотрены наиболее типичные из них.)

`-v` (подробно)

Полезный вариант вызова с ключом `-v`, или verbose mode (многословный, подробный режим), предписывающий программе tar сообщать имена всех архивируемых или разархивируемых файлов. Этот режим позволяет создавать полный список всех сохраняемых или восстанавливаемых файлов. Если резервируется вся система, то получится очень большой список и в нем будет трудно увидеть ошибки.

`-c` (создать новый архив)

Режим `-c` применяется для создания нового архива tar. Если иное не указано, этот ключ инициирует сохранение всей информации на ленточном устройстве, указанном в переменной окружения `$TAPE`. Для сохранения всей системы надо указать tar рекурсивно архивировать все файлы начиная с корневого каталога:

```
# tar -c * /
```

В ответ на ленточном накопителе должен загореться индикатор. Если на ленте достаточно места, в конечном итоге на ней будет сохранена вся система.

Конечно, многие современные диски обладают значительно большей емкостью по сравнению с лентами. Часто их емкость измеряется многими гигабайтами. Поэтому часто имеет смысл сохранять лишь отдельные час-

ти системы. Так, если все файлы, которые подвергались изменениям, находятся в */home*, */usr/local* и */var*, то именно эти каталоги можно указать в командной строке:

```
# tar -c /home /usr/local /var
```

`-t` (перечислить все файлы архива)

В режиме `-t` выводятся имена всех файлов, находящихся в архиве. После завершения работы накопителя можно применить этот ключ, чтобы отобразить содержимое ленты:

```
# tar -t
COPYRIGHT
bin/
bin/cat
bin/chio
bin/chmod
bin/cp
bin/date
bin/dd
bin/df
...
```

В этом выводе видно, что в каждом имени файла отсутствует первый символ «/». Например, файл */COPYRIGHT* представлен как «COPYRIGHT». Эта особенность становится важной при восстановлении файлов, которое будет рассмотрено при описании следующего ключа, `-x`.

`-x` (извлечь все файлы из архива)

В режиме `-x` tar извлекает файлы из архива и копирует их на диск.

tar извлекает файлы в текущий каталог; чтобы перезаписать существующий системный каталог */etc*, надо прежде всего перейти в корневой каталог. Например, для восстановления каталога */etc* в каталоге */home/mwlucas/etc* надо выполнить следующие команды:

```
# cd /home/mwlucas
# tar -x etc
```

Помните замечание о том, что отсутствие первого символа «/» окажется важным? Теперь ясно, почему. Может потребоваться восстановить систему в другом месте на диске, а не в первоначальном. Вся информация со старых серверов может быть сохранена; они могут быть выключены, а их жесткие диски – выброшены или даже подарены. Если бы в именах архивируемых файлов первым символом был «/», tar предположил бы, что имя каждого файла – это абсолютный путь, и восстановил бы их в точном соответствии с их первоначальным расположением; */kernel* с резервной копии был бы восстановлен поверх текущего системного каталога */kernel*! Это было бы плохо.

При отсутствии первого «/» в резервной копии tar восстановит файл в текущем каталоге. Если сохраненные файлы необходимо восстановить со-

гласно их первоначальному расположению, надо набрать `cd /`, чтобы перейти в корневой каталог системы, и затем запустить tar.

`-d` (найти различия в файлах)

Наконец, резервную копию следует проверить с помощью ключа `-d` (diff). Если все файлы на ленте соответствуют файлам в системе, то команда `tar -d` будет выполнена «молча». Однако абсолютное соответствие будет удивительно: по крайней мере, при проведении резервного копирования растут файлы протоколов, поэтому их нельзя сопоставить с сохраненными файлами. Или, если в системе есть активная база данных, ее файлы могут не соответствовать сохраненным.

Необходимо определить, какие ошибки допустимы, а какие надо устранить. Например, можно остановить работу базы данных перед началом резервного копирования и в то же время не заботиться о файлах протоколов. (При наличии трудностей с проверкой резервных копий следует обратиться к документации на программу, вызвавшую эти неполадки.)

Другие ключи tar

Есть и другие полезные ключи, которые можно добавить к только что описанным, чтобы повысить эффективность их применения.

`-z` (gzip)

Ключ `-z` вызывает обработку файлов программой сжатия gzip – как при архивировании, так и при разархивировании. (Сжатые файлы tarballs обычно имеют расширение *.tar.gz* или *tgz*, редко *.taz*.) Сжатие файлов значительно уменьшает размер архива; на самом деле многие резервные копии можно сжать на 50% и более. Все современные версии tar, в отличие от старых, поддерживают `-z`. Поэтому, если необходимо, чтобы сжатые файлы мог прочесть абсолютно каждый, применять ключ `-z` не надо.

`-Z`

В противоположность предыдущему ключу, все версии tar на всех версиях UNIX могут сжимать файлы с помощью ключа `-Z`, который вызывает compress (1). Программа compress не так эффективна, как gzip, но она все-таки уменьшает размер файла. Файлы tarballs, сжатые с помощью `-Z`, имеют расширение *.tar.Z*.

`-y`

tar в составе FreeBSD поддерживает сжатие с помощью программы bzip, более эффективной, чем gzip, но полученные файлы можно прочитать на немногих платформах. Кроме того, bzip требует больше времени процессора, чем gzip. Если сохраняемые файлы будут читаться только на системе FreeBSD, задавайте этот ключ.

`-f`

Ключ `-f` позволяет задать устройство или файл в качестве места для создания архива. Во всех предыдущих примерах это место задавалось переменной `$TAPE`. Если эта переменная не установлена, накопитель на ленте можно указать с помощью `-f`:

```
# tar -cz -f /dev/east0 *
```

Вместо записи резервной копии на ленту можно создать tar-файл. Исходный код, распространяемый по Сети, часто помещается в tar-файлы или tarballs. Для сохранения резервной копии в файле следует также указывать ключ `-f`. Так, для сохранения глав этой книги я время от времени создавал tarball-файл *mybookbackup.tgz*:

```
# tar -cz -f mybookbackup.tgz /home/mwlucas/absolutebsd/
```

После этого tarball можно передать по FTP на сервер; если сгорит дом, книга будет сохранена. Далее можно задействовать телефонный канал и линию электропередачи соседа, загрузить tarball на лэптоп, запустить `tar -xzf mybookbackup.tgz` и работать в окружении обуглившейся древесины, ожидая представителя страховой компании. (Вряд ли я смог бы сделать намного больше в подобной ситуации!)

`-v` (подробно)

Для наблюдения за работой tar применяется ключ `-v`. В этом режиме tar покажет имена всех задействованных файлов. Будьте осторожны: при резервировании всей системы на экран будет выведен большой объем информации.

И еще

У tar есть масса других ключей. Вот некоторые из них: `-C` (изменить каталог), `-p` (восстановить информацию о правах доступа) и `-l` (оставаться в пределах локальной файловой системы[1]). Для получения подробной информации обратитесь к tar(1). Впрочем, для начала работы достаточно сведений этого раздела.

dump/restore

dump(8) – инструмент для проведения резервного копирования, работающий на уровне дисковых блоков. В какой-то мере dump похож на tar. Существенное различие между ними состоит в том, что программе dump есть дело до базового диска – фактически dump напрямую обращается к диску; файловая система не принимается во внимание. Более подробно файловые системы будут обсуждаться в главе 16. Сейчас надо понять, что файловая система задает порядок, в котором единицы и нули располагаются на физическом жестком диске. В различных операционных системах данные размещены по-разному: в Windows есть файловая система NT (NTFS), в Linux – вторая расширенная файловая система (EXT2), а во FreeBSD – файловая система UNIX (UFS). Инструменту, подобному dump, нет дела до того, что за операционная система расположена на физическом диске, – он просто копирует единицы и нули. Это позволяет создавать более надежные резервные копии. Скорее всего, на-

[1] Не захватывать подкаталоги, которые являются точками монтирования других файловых систем. – *Примеч. науч. ред.*

чинающие системные администраторыы хуже знакомы с dump, чем с tar, но dump эффективнее и безопаснее. Когда есть выбор, надо применять dump.[1]

С другой стороны, dump работает с файловыми системами, а не с отдельными файлами. Поэтому dump не подходит для сохранения */etc*, если не надо сохранять весь раздел /. (Тем не менее в этом случае файлы можно будет восстанавливать по отдельности.)

Положительная особенность dump: для сохранения и восстановления информации dump применяет различные программы (восстановление будет рассмотрено в разделе «Восстановление из архива» далее в этой главе). Это означает, что ключи уже нельзя перепутать: при восстановлении данных файл в системе не запишется случайно поверх файла в архиве. Кроме того, dump работает значительно быстрее, чем tar.

Управление работой dump

Возможно, самое большое преимущество dump в том, что она предоставляет пользователям возможность управлять своей работой. Например, они могут указать, какие файлы не надо включать в дамп, и они не будут копироваться. У многих пользователей есть файлы, не представляющие для них значения. Они с радостью согласятся не сохранять эти файлы при условии, что важные для них файлы будут сохранены.

Для установки флага `nodump` воспользуйтесь chflags(1):

```
# chflags nodump filename
```

Содержимое каталога, к которому применен этот флаг, сохраняться не будет. Команда chflags позволяет избежать копирования каталога с дистрибутивами из Интернета. В процессе резервирования это сохранит время и пространство, поскольку эти файлы всегда можно загрузить заново.

Уровни резервирования

Программа dump имеет множество ключей; основной ключ задает *уровень резервирования* (*dump level*), который варьируется от 0 до 9. По умолчанию принимается уровень 0, что подразумевает копирование всех файлов на диске, для которых не установлен флаг `nodump`. Более высокие уровни резервирования означают следующее: «сохранять файлы, которые были изменены или созданы с момента выполнения дампа более низкого уровня». Такая последовательность уровней позволяет выполнять инкрементные дампы – достаточно указать необходимый уровень резервирования как ключ командной строки:

```
dump -0
```

[1] Некоторые системные администраторы не согласны с этим. Они настаивают, что tar(1) лучше. Такое расхождение во мнениях характерно для всего сообщества UNIX. Любые рекомендации, которые я дам, несомненно, вызовут гнев у 49% администраторов, отдавших свои симпатии другому инструменту. (Остальные 2% настаивают, что резервное копирование необходимо выполнять одним из десятка других способов.)

Эта команда означает выполнение дампа уровня 0.

Например, пусть каждый понедельник выполняется дамп уровня 0. Во вторник можно выполнить инкрементный дамп уровня 1. При этом будут сохранены только те файлы, которые изменились с момента проведения дампа в понедельник. Если в среду выполнить дамп уровня 3, то будут сохранено все, что изменилось со вторника. Однако если во вторник выполнить еще один дамп уровня 1, будет сохранено все, что изменилось с понедельника.

Хотя с помощью dump можно выполнять инкрементное резервирование, лучше проводить дампы уровня 0 – просто потому, что тогда будет намного легче восстанавливать файлы, чем в случае серии инкрементных дампов. Дампы уровня 0 выполняются дольше инкрементных дампов и требуют больше места, однако в большинстве случаев время, сбереженное при восстановлении данных, важнее, чем стоимость ленты. При надлежащем планировании дампы уровня 0 можно выполнять ночью.

Другие ключи dump

Работу dump можно полностью контролировать с помощью различных ключей.

`-f`

К сожалению, dump не распознает переменную `$TAPE` и склонен по умолчанию использовать */dev/sa0*. Если это и есть ваш накопитель на ленте, все в порядке. Если нет, то с помощью ключа `-f` можно указать, где следует разместить архив.

Примечание

–f может указывать на файл, а не только на ленточное устройство. Если необходимо поэкспериментировать с dump или скопировать архив на другую машину, сохранение в файле – совершенно правильный выбор.

`-a`

Это другой важный ключ. Несмотря на то что dump старается разумно использовать пространство на ленте, емкость лент значительно выросла с момента создания этой программы и алгоритм, по которому dump вычисляет емкость ленты, не подходит для современных больших лент. Dump ориентируется на старые ленты небольшой емкости и не понимает, что могут быть ленты емкостью 200 Гбайт. В действительности, указав ключ `-f`, можно обойтись вообще без ленты.

Если указан ключ `-a`, то программа не вычисляет необходимую емкость ленты, а выполняет дамп до достижения физического маркера конца ленты. Всегда задавайте этот ключ при дампе в файл.

`-h`

Системный администратор может применять этот ключ, чтобы указать, в каких случаях копировать файлы с флагом `nodump`. В качестве аргумента при задании этого ключа выступает уровень резервирования.

По умолчанию файлы с установленным флагом `nodump` будут сохраняться при каждом дампе уровня 0. На уровне 1 и выше флаг `nodump` будет приниматься во внимание. Ключ `-h` изменяет этот порядок – указывается минимальный уровень, начиная с которого надо учитывать флаг `nodump`. Любой дамп с уровнем ниже заданного будет архивировать все файлы вне зависимости от этого флага.

Например, пусть резервирование системы проводится командой `dump -0 /`; при этом выполняется полный дамп всей системы, в том числе файлов с установленным флагом `nodump`. Если резервные копии заполняют всю ленту или ее не хватает, команду можно изменить так: `dump -0 -h0 /`. В этом случае при резервировании будет учитываться флаг `nodump`, а значит, уменьшится размер резервных копий. На какое-то время дышать станет легче; резервированием можно управлять, и важные данные не будут потеряны. Вслед за этим нужно немедленно заказать новые ленты.

`-u`

В файл */etc/dumpdates* записывается все, что было сохранено в системе. Задав ключ `-u`, этот файл можно обновить. Это наиболее полезно при инкрементном резервировании, когда для успешного восстановления системы необходимо знать дату последнего полного сохранения.

Чтобы выполнить дамп корневого раздела на ленточный накопитель IDE, пропустить файлы с установленным флагом `nodump`, обновить */etc/dumpdates* и извлечь ленту после завершения операции, надо ввести:

```
# dump -0ua -h0 -f /dev/east0 /
```

После этого на экране появятся несколько самоочевидных сообщений, информирующих о выполнении дампа. Если размер дампа превышает размер ленты, dump сообщит о необходимости ее замены.

Тома

Зачастую емкость жестких дисков больше, чем лент; для сохранения данных одного диска может потребоваться несколько лент. Если для резервного копирования одного диска требуется несколько лент, то каждая лента получает номер тома. Первая лента становится томом 1, вторая – томом 2 и т. д.

Dump сообщает номер тома для каждой ленты по мере ее заполнения. (Настоятельно рекомендуем маркировать каждую ленту после завершения операции!) При восстановлении данных либо появится подсказка (номер тома), либо будет задан вопрос, с какого тома следует загружать данные.

Восстановление из архива

Архивы – это замечательно, но они бесполезны, если с их помощью нельзя восстановить систему. Утилита для восстановления данных, restore(8), позволяет извлекать либо целые файловые системы, либо отдельные файлы. Как и в случае с tar и dump, задание ключа `-f` позволяет выбрать устройство или файл, из которого будут восстановлены данные.

Проверка содержимого архива

Для вывода списка файлов, находящихся в архиве, следует указать ключ -t. Если за ним следует имя файла, утилита restore проверит, существует ли такой файл в архиве:

```
# restore -f /dev/ast0 -t /etc/motd
Dump   date: Thu Mar 22 13:30:39 2001
Dumped from: the epoch
Level 0 dump of / on turtledawn.blackhelicopters.org:/dev/ad0s2a
Label: none
     18935      ./etc/motd
#
```

В этом примере для доступа к накопителю на ленте задействован файл устройства */dev/ast0*. Из раздела «Файлы устройств» этой главы мы знаем, что это означает доступ к накопителю IDE. Применение этого файла устройства подразумевает перемотку ленты после завершения операции. Ключ -t указывает, что restore должна проверить наличие файла */etc/motd*. Утилита restore сообщает, когда был выполнен дамп, на какой системе, а также какое дисковое устройство подверглось резервированию.

Извлечение данных из архива

Если известно, что тот или иной файл есть в архиве, то данные из архива можно получить двумя способами: или извлекать файл за файлом, или восстановить всю файловую систему.

Восстановление файла

Если надо извлечь лишь некоторые данные, следует задать ключ -x и имя файла. В этом случае будет извлечен только указанный файл. Например, для восстановления */etc/namedb* из архива, записанного в файл, надо набрать следующую команду (обратите внимание на отклик системы):

```
# restore -f /var/tmp/slashbackup -x /etc/namedb
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
(До сих пор вы не прочитали ни одну ленту. Если вы не знаете, на каком томе
находятся файлы, необходимо начать с последнего тома и двигаться к первому.)
Specify next volume #:
(Укажите следующий том)
Specify next volume #: 1
set owner/mode for '.'? [yn] y
(установить владельца/права доступа для '.'?)
#
```

Примечание

Следует заметить: restore запрашивает номер тома. Если идет восстановление из файла, это неважно. Однако при восстановлении с лент надо указать номер ленты. Если лента одна, введите 1.

После завершения приведенной команды в текущем каталоге должен появиться каталог *etc*, содержащий весь каталог *namedb*.

Восстановление файловой системы

Восстановить всю файловую систему достаточно просто, надо лишь помнить, что лучше не восстанавливать файловую систему поверх существующей. Если требуется серьезное восстановление, то безопаснее удалить раздел и начать все заново.

В следующем примере будет удален раздел на втором диске IDE, а затем данные будут восстановлены с резервной ленты. Работа с диском здесь не будет подробно рассматриваться (дополнительную информацию см. в главе 13), однако все действия в общем виде можно представить так:

1. Создание новой файловой системы с помощью fdisk.
2. Присоединение этой файловой системы к дереву каталогов, в */mnt*.
3. Переход в этот каталог и запуск restore с указанием ленточного устройства */dev/ast0*.

Вот эти команды:

```
# newfs /dev/ad1s1g
# mount /dev/ad1s1g /mnt
# cd /mnt
# restore -rf /dev/ast0
```

Интерактивное восстановление

Одна из наиболее интересных особенностей dump – интерактивный режим, `-i`, позволяющий «вскрыть» дамп (файла или ленты) и получить доступ к нему с помощью какого-либо инструмента командной строки. При этом можно указать файлы, которые необходимо восстановить. Интерактивный режим бывает весьма кстати, когда пользователь говорит примерно следующее: «Я случайно удалил файл с резюме. Он лежал в моем домашнем каталоге, а в его имени было слово «resume». Точного имени не помню. Можно его восстановить?» Ясно, что ключ `-t` не поможет; точное название файла неизвестно! Вместо этого можно перейти в интерактивный режим и поискать этот файл.

Следующий листинг демонстрирует интерактивное открытие файла дампа *root.dump*. (Будьте уверены, этот режим так же хорошо работает и с дампом на ленте!) Restore предоставляет *командную строку restore* (*restore command prompt*), которая выглядит подобно обычной командной строке FreeBSD, но поддерживает только команды, специфичные для restore.[1]

```
#restore -i -f root.dump
restore > ls
```

[1] Команды, которые можно набрать, вызвав restore, называются так же, как команды FreeBSD. Указанные ниже ссылки на команды ls(1) и cd(1) на самом деле неверны, т. к в разделе 1 Руководства описаны команды ls и cd самой ОС FreeBSD, а не встроенные команды restore. – *Примеч. науч. ред.*

```
.:
.cshrc          compat@         kernel.GENERIC  modules.good/   sys@
.profile        dev/            kernel.good     modules.old/    tmp@
COPYRIGHT       dist/           kernel.old      proc/           usr/
bin/            etc/            laptop-kernel   root/           var/
boot/           home@           mnt/            sbin/
cdrom/          kernel          modules/        stand/

restore >
```

После открытия дамп-файла (как показано в листинге) по нему можно продвигаться, используя ls(1) для вывода содержимого каталога и cd (1) для перехода в другие каталоги.

Найдя необходимый файл, его надо восстановить. В интерактивном режиме restore хранит список всех файлов, которые необходимо извлечь из дампа. Каждый файл добавляется в список, а затем restore восстанавливает из дампа все указанные файлы. Для добавления файла в список следует набрать add и имя файла. Так, для восстановления */etc/master.passwd* и */etc/passwd* в интерактивном режиме из дампа, показанного в предыдущем примере, надо ввести следующие команды:

```
restore > cd etc
restore > add master.passwd
restore > add passwd
restore > extract
You have not read any tapes yet.
Unless you know which volume your file(s) are on you should start
with the last volume and work towards the first.
Specify next volume #: 1
set owner/mode for '.'? [yn] y
restore > quit
```

Выражение «volume #» в одном из предшествующих листингов – это номер ленты, которая помещена в машину. Для многих дампов необходимо несколько лент, и каждая из них во время выполнения дампа получает номер тома. При восстановлении из файла номер тома равен 1. Если бы весь дамп можно было разместить на одной ленте, то номер тома также был бы равен 1.

Примечание

Когда бы ни проводилось полное восстановление диска, перед следующим после него инкрементным дампом необходимо выполнить дамп уровня 0. Restore изменяет положение данных на диске, поэтому будущие инкрементные резервирования будут бесполезны без нового дампа уровня 0. Кроме того, нужно помнить, что при выполнении полного резервирования жить будет намного легче.

Запись происходящих событий

Script (1) – один из редко упоминаемых, но довольно полезных инструментов, о котором должен знать каждый системный администратор. Он записы-

вает все набранные команды, а также все, что появляется на экране. Запись происходит в файл *typescript*. Этот файл можно использовать для записи ошибок или большого вывода команд, который можно проанализировать впоследствии. Script продолжает запись до тех пор, пока не будет дана команда `exit`.

Например, если программа всякий раз спотыкается на одном и том же месте, script можно задействовать для копирования набранных строк и ответов, появляющихся на экране. Это особенно полезно при обновлении системы или компоновке программного обеспечения из исходного кода; последние 30 строк (или около того) файла typescript представляют собой прекрасное дополнение к письму о помощи.

Управление версиями

Вообще говоря, управление версиями (revision control) – это процесс отслеживания изменений. В мире UNIX под этим понимаются изменения в исходном коде и в файлах конфигурации. Управление версиями позволяет разработчику видеть, как некий фрагмент кода выглядел в определенный момент. Администратор может представить, как была сконфигурирована система до возникновения неполадок. Управление версиями может быть полезно даже скромному писателю – он может увидеть, как менялась рукопись с течением времени. Если не применять этот механизм, работа будет труднее, чем ей положено быть.[1]

Несмотря на то что существует много систем управления версиями, от системы управления исходным кодом SCCS (Source Code Control System) в UNIX до Visual SourceSafe компании Microsoft, здесь будет рассмотрена система управления версиями RCS (Revision Control System), входящая почти во все системы UNIX. После изучения RCS работать с большинством других систем управления версиями будет просто.

При управлении версиями, по существу, записывается все, что происходит с тем или иным файлом. Во-первых, файл помечается как *захваченный* (*checked out*). Это знак системе, что файл будет изменяться. Далее файл можно редактировать, изменения будут записываться в системе, а затем файл можно передать другим пользователям для последующего редактирования. Для поддержки этих действий RCS применяет три команды: ci (check-in, регистрация), co (check-out, захват) и rcs.

Управление версиями напоминает обычную, старую добрую библиотеку. Прежде всего, надо указать системе RCS на необходимость отслеживания файла – поместить его в библиотеку. Для того чтобы работать с этим файлом, его необходимо получить – подобно книге из библиотеки. Если файл

---

[1] Одни рецензенты отметили, что этот раздел может отпугнуть начинающих сисадминов. Другие сожалели, что не знали об управлении версиями в начале своей карьеры. Не пугайтесь: этот материал не так уж и сложен; он облегчит вашу жизнь.

выдан, то никто иной не может его сохранять или редактировать, хотя любой законный пользователь может просматривать, копировать, компилировать этот файл – обращаться к нему. После завершения работы с файлом он сдается обратно (check-in) и становится доступен для редактирования другими пользователями. Весь процесс называется RCS.

В RCS каждый файл имеет номер версии. Всякий раз, когда отредактированный файл возвращается в систему, Revision Control System сравнивает возвращаемый файл с захваченным. При наличии хоть каких-то изменений номер версии увеличивается на единицу, – таким образом система отслеживает изменения в файле. Номер версии можно применять для идентификации отдельных версий файла.

Процесс управления версиями начинается с регистрации файла с помощью ci(1), что напоминает сдачу книги в библиотеку. Например, хороший кандидат для защиты с помощью RCS – файл */etc/rc.conf*. Для начала надо ввести `ci <path/filename>`, как показано в следующем листинге:

```
❶ # ci /etc/rc.conf
rc.conf,v <-- rc.conf
❷ enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
❸ >>System configuration file
❹ >> .
initial revision: 1.1 done
#
```

При первой регистрации файла с помощью ci (❶) эта команда создает или редактирует файл управления версиями. Это показано во второй строке: создается файл *rc.conf,v*. Далее система запрашивает описание (❷), и его надо ввести (❸), чтобы каждый пользователь RCS позднее мог понять, что это за файл. (Это описание не настолько важно для стандартных системных файлов, но оно будет очень полезно для исходного кода или файлов конфигурации самостоятельно разработанных или сложных программ.) После того как описание набрано, для завершения работы ci следует ввести точку на отдельной строке (❹).

Если сразу после регистрации файла запустить ls, то можно увидеть, что файл как будто исчез. Вместо этого появится файл с тем же именем, но с замыкающими символами «,v». Это файл RCS, в котором хранятся сам файл и его изменения. Для некоторых файлов такой способ очень удобен, но файлы с исходным кодом или веб-страницами не могут просто исчезнуть. Чтобы решить эту проблему, при регистрации файла нужно оставить его копию в рабочем каталоге. Для этого надо запустить `ci -u`.

Если файл исчез, а его копию необходимо поместить в рабочий каталог без редактирования файла, надо выполнить команду `co`. В следующем примере файл *test* был извлечен из файла *test,v*. Номер версии – 1.1.

```
# co test
test,v  -->  test
```

```
revision 1.1
done
#
```

Обратившись к каталогу, в котором находится файл *test*, можно увидеть:

```
# ls -l
total 62
-r--r--r--  1 mwlucas  mwlucas  12663 Oct  4 18:06 test
-r--r--r--  1 mwlucas  mwlucas  12867 Oct  4 17:56 test,v
#
```

Владелец этого файла – пользователь mwlucas, однако права доступа – «только для чтения» (`-r--r--r--`; см. главу 7). У пользователя mwlucas нет права редактировать свои файлы! Дело в том, что файл не захвачен пользователем mwlucas, хотя он зарегистрировал этот файл и передал его библиотекарю Revision Control System. Он может просматривать этот файл, но для его редактирования об этом нужно попросить библиотекаря. Файл необходимо захватить, а затем блокировать. Для этого предназначен ключ `-l` команды `co`.

```
# co -l test
test,v  -->  test
revision 1.1 (locked)
done
```

Обратите внимание на третью строку этого листинга (вторая строка вывода), в которой указано «locked» (заблокирован). Этот файл захвачен и заблокирован. Только пользователь, выполнивший блокировку, может сохранять этот файл (пока он его не разблокирует).

Запустив после этой операции ls еще раз, можно увидеть, что права доступа к файлу *test* снова разрешают чтение и запись, а значит, этот файл можно сохранять.[1] (Права доступа рассматриваются в главе 7.) Любой, кто попытается захватить этот файл, получит предупреждение о том, что файл используется. При этом будет указано имя того, кто этот файл заблокировал.

После завершения работы с файлом его можно зарегистрировать и снять блокировку, чтобы другие пользователи могли его редактировать. Для этого надо выполнить команду `ci -u`.

```
# ci -u test
test,v  <--  test
new revision: 1.2; previous revision: 1.1
```

[1] Предупреждение для пользователей vi: если вы или ваша группа владеете файлом, команда `w!` вызовет изменение прав доступа и позволит вам осуществлять запись в файл даже без его регистрации. Все будет выглядеть замечательно, однако следующий пользователь, который зарегистрирует файл, перезапишет ваши изменения! Будьте осторожны с этой командой; если vi говорит, что у вас нет прав на сохранение файла, это весомое предупреждение, к которому лучше прислушаться.

```
❶ enter log message, terminated with single '.' or end of file:
>> enable sendmail again
>> .
done
#
```

При регистрации файла система запросит сообщение, которое будет записано в протоколе (❶). Здесь надо ввести краткое описание внесенных изменений. (Такие протокольные сообщения (log messages) сравнимы с сообщениями CVS, которые можно увидеть в различных почтовых рассылках BSD.)

Благодаря сообщениям, хранимым в протоколе, другие пользователи легко узнают обо всех сделанных изменениях или могут увидеть, какие *попытки* предпринимались при ошибочных изменениях, когда кому-либо приходилось начать отладку. Протоколы RCS могут быть полезны и владельцу файла, спустя несколько месяцев удивленно пытающемуся вспомнить, о чем же он думал в то время.

Примечание

Если в систему RCS помещается много файлов, то файлы «,v» могут быстро заполнить каталог. Эти файлы можно спрятать, создав каталог с именем «RCS». Тогда программа ci поместит в него все файлы «,v», разгрузив рабочий каталог.

Теперь, когда ясны основы регистрирования и захвата файлов, исследуем более интересные функции RCS: получение старых версий файлов, снятие блокировок, нахождение различий между версиями файлов и размещение в файлах идентификаторов RCS.

Получение старых версий

Каждый файл в RCS имеет номер версии, который увеличивается при каждой регистрации файла. Впрочем, система помнит, как выглядел файл ранее, поэтому номер версии позволяет захватить любую предыдущую версию файла.

Например, пытаясь отследить только что обнаруженную ошибку, можно извлечь более ранние версии кода и увидеть, проявлялась ли она в них. Для этого команду `co` надо запустить с ключом `-r`. Вот как извлекается версия 1.1 файла */etc/rc.conf*:

```
# co -r1.1 rc.conf
RCS/rc.conf,v  -->  rc.conf
revision 1.1
done
#
```

Снятие блокировок

Всегда регистрируйте файлы после завершения работы с ними. В противном случае пользователю, желающему редактировать заблокированный файл, придется снять блокировку, и все изменения, внесенные предыдущим пользователем с момента блокировки, будут потеряны.

Для снятия блокировки с файла применяйте `rcs -u`. RCS попросит ввести сообщение о причине снятия блокировки, и это сообщение будет послано пользователю, заблокировавшему файл.

Примечание

При снятии блокировки будьте осторожны: если кто-либо в это время действительно редактирует файл, то он, по понятным причинам, будет раздосадован. Другое дело, если он отправился домой на целый день. Необходимо приложить все усилия, чтобы найти этого человека, прежде чем снимать его блокировку!

Просмотр протокольных сообщений

Команда rlog показывает протокольные сообщения для того или иного файла.

```
# rlog /etc/rc.conf

RCS file: /etc/RCS/rc.conf,v
Working file: /etc/rc.conf
head: 1.4
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 4;     selected revisions: 4
description:
----------------------------
revision 1.4
❶date: 2000/09/08 17:45:29;  ❷author: mwlucas;  ❸state: Exp;  ❹lines: +2 -0
minor updates
----------------------------
revision 1.3
date: 2000/09/07 19:05:30;  author: mwlucas;  state: Exp;  lines: +1 -1
❺ *** empty log message ***
----------------------------
revision 1.2
date: 2000/09/05 16:09:47;  author: mwlucas;  state: Exp;  lines: +1 -1
enable sendmail
----------------------------
revision 1.1
date: 2000/09/02 14:53:43;  author: mwlucas;  state: Exp;
Initial revision
----------------------------
=============================================================================
#
```

Здесь представлена вся полезная информация, включая дату каждой регистрации (❶), автора изменения (❷), состояние (❸), которое не будет здесь обсуждаться (см. ci(1)), а также количество измененных строк (❹).

Просмотр истории версий файла

Следует отметить, что в предыдущем разделе не было набрано сообщение для версии 1.3 (❺). Сделанные изменения можно просмотреть с помощью команды rcsdiff, имеющей три аргумента – две версии и имя файла:

```
# rcsdiff -r1.2 -r1.3 /etc/rc.conf
===================================================================
RCS file: /etc/RCS/rc.conf,v
retrieving revision 1.2
retrieving revision 1.3
diff -r1.2 -r1.3
6c6
< inetd_enable="NO"
❶ > inetd_enable="YES"
#
```

Очевидно, что в период между созданием версий был активизирован inetd (❶). Это важная информация (особенно если речь идет о рабочей системе и тем более, если ее администрируют несколько человек). Программа inetd рассмотрена в главе 13.

К тому же при запуске rcsdiff можно задавать произвольные номера версий. Это позволяет просмотреть изменения, выполненные в период между созданием любых двух версий. Таким образом можно затребовать сведения о различиях между версиями 1.1 и 1.4 и увидеть все изменения, сделанные с момента первой регистрации файла.

Строки Ident и ident

Строки идентификации позволяют упростить просмотр сведений о том, кто и когда изменял тот или иной файл. Так, если последнюю неделю сервер ведет себя странно, полезно знать, что изменилось неделю назад. Конечно, можно запустить rlog для каждого конфигурационного файла системы и проанализировать изменения, но это немного утомительно. Куда лучше просто взглянуть на файл, в котором представлена вся необходимая информация. Вот где нужны строки идентификации. Эти строки можно поместить в конфигурационные файлы системы. Когда пользователь захватывает файл, RCS автоматически обновляет эти строки.

RCS поддерживает много строк идентификации, каждая из которых представлена в форме `$string$`. Например, в строке `$id$` размещается информация о последнем изменении в файле. Всегда будет полезно разместить `#$Id$` в первой строке системного файла */etc/rc.conf* и зарегистрировать его. Строка будет такой:

```
#$Id: rc.conf,v 1.5 2000/10/05 18:29:49 mwlucas Exp mwlucas $
```

Примечание

Знак «решетки» (#) – это комментарий, который предписывает */etc/rc* не выполнять эту строку при запуске. Знак комментария может быть любым подходящим для данного файла.

Далее представлены типичные строки идентификации.

Строка идентификации `$Id$` встречается чаще всего. Она показывает полный путь к файлу RCS, номер версии, дату и время первой регистрации, автора, состояние и пользователя, выполнившего последнюю блокировку.

Примечание

Состояние RCS – это произвольная строка, которую можно назначить с помощью ci(1) или rcs(1). Для файла можно задать произвольные состояния, которые позволяют понять назначение файла и условия его использования. Многие люди применяют эту строку, чтобы пометить файл как «экспериментальный» или «рабочий» или указать: «Ничего не меняйте ни по каким причинам». Беспокоиться о состоянии файла не надо, достаточно принять значение по умолчанию Exp («experimental», экспериментальный). Обычно в системном администрировании строка состояния RCS не применяется.

`$Header$` – другая распространенная строка идентификации. Она сходна с `$Id$`, за исключением того, что она сообщает полный путь к файлу RCS, а не просто имя файла.

`$Log$` – интересная строка идентификации, добавляющая протокольные сообщения RCS к самому файлу; при просмотре файла эти сообщения можно увидеть. Они могут переполнять интенсивно редактируемые файлы, но могут быть полезными для файлов, которые изменяются реже. Так, */etc/rc.conf* на рабочей системе может вообще не изменяться месяц. Если описываемую строку идентификации поместить в этот файл, в нем можно будет просматривать все протокольные сообщения RCS. Сразу станет ясно, что изменено, кем и почему.

Есть несколько других строк идентификации, но они, по сути, являются подмножеством трех описанных строк. См. ident(1) для просмотра полного списка.

Идем дальше

Управление версиями – это мощный инструмент. Достаточно полное пособие по этой теме можно найти по адресу *http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/rcs/*.

Однопользовательский режим

В отличие от других операционных систем, BSD-системы могут выполнять минимальную начальную загрузку. Это важно для выявления неполадок и восстановления системы.

Минимальная начальная загрузка, так называемый однопользовательский режим, подразумевает загрузку ядра и нахождение устройств, но при этом не происходит автоматического монтирования диска, подключения к сети, активизации защиты системы и запуска стандартных служб UNIX. Однопользовательский режим – это первый рубеж, на котором система, возможно, выдаст приглашение на ввод команды (command prompt). Отсюда можно активизировать любые из этих служб.

В начале загрузки система FreeBSD выполняет десятисекундный обратный отсчет времени, предоставляя возможность приостановить загрузку. При нажатии любой клавиши будет выдано приглашение OK. Для загрузки в однопользовательском режиме введите такую команду:

```
OK> boot -s
```

На экране одно за другим будут появляться сообщения об опрашивании устройств, а затем система предоставит возможность выбрать оболочку (shell). (В корневом разделе можно запустить любую оболочку; обычно применяют оболочку по умолчанию, */bin/sh*, однако при желании можно использовать */bin/tcsh*.)

Однопользовательский режим можно применять для переустановки пароля root. Для этого надо выполнить команду:

```
# fsck -p
# mount -a
# passwd root
# exit
```

Команда mount -a монтирует файловые системы, представленные в */etc/fstab* (см. главу 13). Если одна из них работает неустойчиво или приводит к аварии системы, то файловые системы можно монтировать по отдельности, указывая их как аргумент mount (например, mount /usr).

Если система повреждена еще больше, то файл */etc/fstab* может не читаться. В этом случае корневой раздел можно монтировать, указав имя устройства (см. главу 13). Вероятно, это либо */dev/ad0s1a* (для дисков IDE), либо */dev/da0s1a* (для дисков SCSI). При монтировании раздела надо задать имя устройства и указать точку монтирования, в которой раздел должен быть доступен. Например, для монтирования первого диска IDE в root надо ввести следующую команду:

```
# mount /dev/ad0s1a /
#
```

Вот случай из моей практики. Однажды на системе FreeBSD я экспериментировал с режимом Linux. Экспериментальная файловая система Linux была прописана в */etc/fstab*. После обновления до последней FreeBSD-current (см. главу 6) эта файловая система перестала работать.[1] Ее монтирование приводило к аварийной ситуации. Хуже того, компьютер пытался монтировать эту файловую систему при каждом запуске. Загрузка происходила частично, прерывалась, а затем начиналась заново, снова и снова и снова. Я загрузился в однопользовательском режиме, вручную запустил fsck для /usr, смонтировал его и с помощью vi отредактировал */etc/fstab*, закомментировав файловую систему, которая являлась причиной неполадок.

---

[1] Пожалуйста, обратите внимание: ни одно из приведенных замечаний не является рекомендацией. Одна из замечательных особенностей FreeBSD состоит в том, что эта система не запрещает выполнять опасные операции; вместо этого она позволяет понять, почему они считаются опасными.

Для разрешения или запрещения запуска чего-либо в ходе загрузки можно воспользоваться похожим методом – отредактировать */etc/rc.conf* (см. главу 9) или соответствующий */usr/local/etc/rc.d* (см. главу 11). Таким образом, к примеру, можно изменять уровень защиты (см. главу 7) до завершения загрузки системы.

Примечание

Подключение к сети в однопользовательском режиме выполняйте при помощи сценария */etc/netstart*. Это позволит избежать запуска сетевых служб, таких как inetd или httpd.

Список команд, доступных в однопользовательском режиме, зависит от того, какие разделы были монтированы. Базовые команды доступны в корневом разделе, в */bin* и */sbin*. Другие (такие как *vi* и *ee*) живут в */usr* и не будут доступны без монтирования этого раздела. Изучите */bin* и */sbin*, чтобы понять, что необходимо для работы.

Если не удается загрузиться даже в однопользовательском режиме, остается последнее: диск восстановления (fixit disk).

Диск восстановления

Наилучший способ изучить UNIX состоит в том, чтобы поиграть с этой системой, и чем труднее игра, тем больше можно изучить. Если игра идет не на шутку, что-нибудь обязательно «полетит». Восстановление неработоспособной системы, возможно, является самой короткой дорогой к получению знаний. Этот раздел предназначен для тех, кто помог системе перестать загружаться или планирует достаточно быстро набраться знаний, чтобы решиться на это. Вам придется научиться многому и быстро, хотя, в основном, и самостоятельно.

Одна из наиболее интересных частей FreeBSD – диск восстановления. Можно начать загрузку с установочного носителя, но вместо установки ОС выбрать *режим восстановления* (*fixit mode*). Для этого надо выбрать первый пункт меню, которое предоставляет инсталлятор.

Для того чтобы успешно применять систему восстановления, *необходимо* в какой-то мере понимать системное администрирование! По существу, диск восстановления предоставляет приглашение для ввода команд и множество утилит UNIX. Для устранения неполадок необходимо задействовать свою голову и проанализировать сообщения об ошибках, выданных во время загрузки. Вы против компьютера. В первых трех случаях из шести, когда я прибегал к помощи диска восстановления, победу одерживал компьютер. Однако время было потрачено не зря, поскольку теперь мне по силам восстановить поврежденную систему. Эту книгу определенно следует прочесть до конца, прежде чем пытаться что-то делать.

Пошаговое изложение процесса восстановления не представляется возможным; последовательность действий целиком зависит от ущерба, причиненного бедному, невинному компьютеру. Не надо отчаиваться – режим восста-

новления поможет избежать полной переустановки системы. Бывало, я случайно уничтожал каталоги */etc* и */dev* или «подвешивал» программу getty, которая отображает приглашения для ввода регистрационного имени (login prompt). Осмотрительное применение режима восстановления помогает найти выход в таких ситуациях за долю того времени, которое бы потребовалось для переустановки системы.

Для использования режима восстановления необходим установочный носитель FreeBSD (CD или две загрузочных дискеты) и либо дискета восстановления, либо второй CD из поставки FreeBSD. Образ диска восстановления можно загрузить с любого FTP-сервера FreeBSD. Компакт-диски с последней версией FreeBSD можно приобрести у разных поставщиков, например у Daemon News и FreeBSD Mall.

Важно, чтобы диск восстановления примерно соответствовал запускаемой версии FreeBSD. Мелкие различия не играют роли, однако попытка восстановить систему 4.4-stable с помощью диска восстановления 5.0-current обречена на неудачу.

Начните загрузку с установочного носителя. В первом меню будет предоставлена возможность выбора режима восстановления. Выберите его. Далее выберите носитель: CD или дискету. Если CD есть, следует предпочесть его. Дискета восстановления содержит только программы, которые можно разместить на одном гибком диске. А на CD записан полный комплект программ, которые есть в комплекте FreeBSD. Хотя на CD может не оказаться любимого редактора или предпочтительной оболочки, на нем есть все, что действительно необходимо.

Существующий жесткий диск можно монтировать в */mnt*, а программы – либо в */stand*, либо в */mnt2*. Список команд, которые появятся в этих каталогах, варьируется от системы к системе; запустите `ls /mnt2` и `ls /stand`, чтобы узнать, что есть в наличии.

В некоторых случаях можно надеяться лишь на то, что удастся смонтировать жесткий диск и считать с него данные. Тут поможет компакт-диск восстановления, содержащий все инструменты для подключения системы к сети. Одно из преимуществ программы установки FreeBSD – возможность сохранения существующих разделов. Работая с диском восстановления, можно с помощью tar заархивировать файлы данных, а затем переустановить систему. Когда система заработает, эти данные можно извлечь из файлов tarballs и вернуть их в систему.

Итак, мы знаем, как восстанавливать систему, конфигурацию и файлы, и можем приступить к подгонке операционной системы под аппаратное обеспечение компьютера.

4

Эксперименты с ядром

Первый шаг в оптимизации FreeBSD – конфигурирование *ядра* (*kernel*). Новичков в администрировании UNIX слово kernel может устрашить. Ведь ядро – это одна из тех скрытых частей системы, которые не предназначены для дилетантского вмешательства простых смертных. На самом деле в некоторых версиях UNIX, таких как Solaris, непосредственные манипуляции с ядром немыслимы. Впрочем, в мире UNIX с открытым исходным кодом вмешательство в работу ядра представляет собою наилучший способ улучшить производительность. (Вероятно, если есть такая возможность, это наилучший способ и для настройки других систем.)

Ядро FreeBSD можно настраивать динамически, или изменять на лету. При необходимости можно изменять большинство аспектов системной производительности. Далее будет обсуждаться интерфейс ядра sysctl и его применение для модификации рабочего ядра.

В то же время, некоторые части ядра не подлежат изменению непосредственно при работе системы, а для настройки некоторых функций ядра необходима значительная реконфигурация. Кроме того, размер ядра можно уменьшить за счет удаления ненужных компонентов. Для этого лучше всего собрать собственное ядро, и далее будет показано, как это сделать.

Наконец, в этой главе будут рассмотрены загружаемые модули ядра – подсистемы ядра, которые можно включать или выключать при необходимости.

Что такое ядро?

Ядро (*kernel*) имеет много разных определений. Значительная их часть абсолютно сбивает с толку. Следующее определение не претендует на полноту, но оно понятно и удовлетворяет нашим целям: *ядро* – это интерфейс между аппаратным и программным обеспечением.

Ядро позволяет записывать данные на дисковые накопители и работать в сети. Оно обслуживает операции процессора и памяти. Оно преобразует MP3 в поток нулей и единиц, которые понимает звуковая карта. Оно указывает мо-

нитору, где отображать маленькие цветные точки. Ядро предоставляет интерфейсы программам, которым необходим доступ к аппаратным средствам.

Описать работу ядра легко (по крайней мере, в этой упрощенной манере), но ее трудно выполнять. При взаимодействии с аппаратными средствами различные программы опираются на различные интерфейсы ядра, а аппаратные средства предоставляют свои ресурсы разными способами. Ядро должно справляться со всем этим. Например, ядро управляет распределением памяти; если программа запрашивает память способом, который не поддерживается ядром, возникает затруднение. (Программы запрашивают память различными способами.) А если ядро не знает, как взаимодействовать с сетевой картой, сетевая карта не будет работать. При загрузке ядро получает сведения об аппаратных средствах, и их дальнейшая работа определяется полученным сведениями. Значит, этим процессом надо управлять. Одни сетевые карты прекрасно распознаются, а другие накрепко блокируются при простом запросе.

Фактически, ядро – это файл на диске: */kernel*. *Модули ядра* (*kernel modules*) – программы ядра, которые можно загружать или выгружать после начальной загрузки, – живут в */modules*. В эпоху съемных аппаратных средств, таких как PC Card и USB, модули ядра необходимы. Кроме того, они могут предоставить дополнительную функциональность, которая не требуется от ядра постоянно. Файлы, находящиеся вне */kernel* и */modules*, не являются частями ядра; эти файлы и программы образуют группу userland[1], которая, как явствует из названия, предназначена для пользователей. В то же время, эти программы и данные обращаются к функциям ядра.

В только что установленной системе можно также увидеть файл */kernel.GENERIC* – стандартное установочное ядро. В системах, уже работающих какое-то время, можно найти другие ядра. Одни из них использовались ранее, другие предназначены для запуска при особых обстоятельствах, третьи ядра – экспериментальные, которые не заработали. Команда FreeBSD стремится к тому, чтобы сделать настройку и установку ядра как можно проще. Давайте посмотрим.

Конфигурирование ядра

FreeBSD предоставляет два способа для настройки существующего ядра: sysctl(8) и загрузчик операционной системы (boot loader).

sysctl

Программа sysctl позволяет взглянуть на параметры ядра и в некоторых случаях установить их значения. Запутаем дело: иногда эти параметры тоже называются *sysctls*.[2] sysctl – это мощный инструмент, поскольку во многих

---

[1] Userland (англ.) – букв. земля пользователя. Согласно файлу жаргона Эрика Реймонда, под userland понимается любой код, не входящий в ядро. – *Примеч. перев.*

[2] Sysctls – множественное число от sysctl. – *Примеч. перев.*

случаях он позволяет увеличивать производительность без повторной сборки ядра или переконфигурирования приложения. К сожалению, это палка о двух концах – таким образом можно нарушить работу программ и сделать пользователей очень несчастными.

Все операции над параметрами настройки ядра (sysctls) выполняются с помощью команды sysctl(8). По ходу изложения материала я буду обращать внимание на ту или иную операцию sysctl, которая изменяет поведение системы, но сначала эти операции необходимо понять.

Прежде чем взяться за sysctls, выясним, какие из них доступны в системе. Следующая команда сохранит список операций в файле, который можно легко изучить:

```
# sysctl -A > sysctl.out
```

После выполнения этой команды файл *sysctl.out* будет содержать сотни параметров настройки ядра и их значений. Большая их часть пока неясна, но некоторые понять нетрудно:

```
kern.hostname: bigbox.blackhelicopters.org
```

Этот sysctl, называемый *kern.hostname*, имеет значение «bigbox.blackhelicopters.org». Система, на которой я запустил эту команду, называется «bigbox.blackhelicopters.org». По названию sysctl нетрудно догадаться, что это параметр ядра, хранящий имя компьютера. Довольно легко, не правда ли?

Есть намного более странные параметры:

```
p1003_1b.memory_protection: 0
```

Как пользователь я понятия не имею о том, что это значит. Все же, если у меня возникают трудности и я обращаюсь за помощью к поставщику системы или в почтовую рассылку, я получаю необходимую информацию. Мне могут посоветовать настроить этот параметр для лучшей поддержки программного обеспечения.

Параметры настройки ядра организованы по древовидной схеме, которая называется базой управляющей информации (Management Information Base, MIB). В ней есть несколько основных категорий, таких как net, vm и kern. (Дерево базы управляющей информации используется и в других задачах по администрированию системы; далее в этой книге будет приведен другой пример.) Каждая категория, в свою очередь, подразделяется на подгруппы; например, категория net, охватывающая все сетевые sysctls, разделена на категории IP, ICMP, TCP и UDP. Термины sysctl MIB и sysctl часто употребляются попеременно. Есть много видов MIB – примеры SNMP MIB будут представлены в главе 19, но в этой главе рассмотрены только базы управляющей информации sysctl.

Ранее был приведен параметр `kern.hostname`, и если взглянуть на sysctls, доступные на машине, можно увидеть, что целая группа параметров начинается с «kern». Все они представляют общие значения ядра. Если продвинуться чуть дальше, можно увидеть целую группу параметров sysctls, начинающихся с «kern.ipc», таких как эти:

```
kern.ipc.maxsockbuf: 262144
kern.ipc.sockbuf_waste_factor: 8
kern.ipc.somaxconn: 128
. . .
```

Эти параметры описывают возможности IPC[1] ядра. Ветвление дерева sysctls может продолжаться на нескольких уровнях.

Со временем обязательно возникнет необходимость в настройке отдельных баз управляющей информации, таких как `net.inet.raw.recvspace`. Каждый MIB имеет значение, которое связано с неким буфером, настройками или особенностями, используемыми ядром. Изменение этого значения повлечет за собой изменение работы ядра. Например, некоторые параметры настройки ядра управляют выделением памяти для каждого сетевого соединения. Если сетевая производительность низка, можно увеличить объем системных ресурсов, резервируемых для сетевых соединений. Корни дерева sysctl MIB представлены в табл. 4.1.

Таблица 4.1. Корни дерева sysctl MIB

Sysctl	Функция
kern	основные функции ядра
vm	виртуальная память
vfs	файловые системы
net	сетевые функции
debug	отладочная информация
hw	информация об аппаратных средствах
machdep	платформо-зависимые параметры (то есть Alpha, i386 и т. д.)
user	информация об интерфейсах userland
p1003_1b	характеристики POSIX[a]

[a] POSIX – это международный стандарт для программ UNIX и функций ядра. Большинство систем FreeBSD соответствуют POSIX.

Каждый sysctl может представлять собой *строку* (*string*), *целое число* (*integer*), *двоичное значение* (*binary value*) либо *неясный код* (*opaque*). String – это текст свободного формата с произвольной длиной; integer – обыкновенное целое число; binary value – либо 0 (off), либо 1 (on); значение opaque представлено в машинном коде, который могут интерпретировать только специальные программы.

К сожалению, параметры настройки ядра недостаточно документированы. Нет отдельного документа, в котором перечислены все имеющиеся базы

[1] IPC – это сокращение от «interprocess communication» (межпроцессная коммуникация). Этот механизм нужен различным программам. В документации по той или иной программе сказано, надо ли изменять этот sysctl.

управляющей информации (sysctl MIB) и их назначение. Но описание каждого параметра присутствует на странице документации по компоненту, которым он управляет. Например, исходная документация для MIB kern.securelevel (более подробно этот sysctl рассмотрен в главе 7) представлена в init(8). Многие параметры вообще не имеют документации. В приложении приведен список распространенных параметров настройки ядра и описано их назначение.

К счастью, некоторые параметры очевидны. Так, если обратиться к файлу с сохраненными MIB, почти в самом верху можно увидеть один из них:

```
kern.bootfile: /kernel
```

Этот MIB важен, если постоянно загружаются разные ядра. (Далее в этой главе будет рассмотрено, как загружать дополнительное ядро.) Если проводится отладка и последовательная загрузка с различных ядер, можно легко выяснить текущее ядро, проанализировав значение этого MIB. Не один раз я загружал тестовое ядро, исследовал неполадки и устранял их, забывая при этом, с какого ядра происходила загрузка.

Чтобы увидеть поддерево MIB, доступных в отдельном дереве, таком как kern, введите такую команду:

```
# sysctl kern
kern.ostype: FreeBSD
kern.osrelease: 5.0-CURRENT
kern.osrevision: 199506
...
```

Этот список может быть длинным. Он может пригодиться, когда необходимо выяснить, какие sysctls представлены в ядре. Чтобы получить только значение отдельного параметра настройки ядра, задайте имя MIB как аргумент:

```
# sysctl kern.securelevel
kern.securelevel: -1
#
```

В этом случае `kern.securelevel` имеет целочисленное значение –1. Смысл этого значения обсуждается в главе 7.

Изменение sysctls

Некоторые параметры sysctls предназначены только для чтения. Рассмотрим MIB-деревья hw (hardware, аппаратное обеспечение) и machdep (machine dependencies, машинно-зависимые параметры).

```
hw.machine: i386
```

Разработчики FreeBSD все еще развивают технологию переноса системы на аппаратные средства PowerPC с помощью программных настроек, поэтому данный параметр доступен только для чтения и его изменение может привести лишь к неработоспособности системы. Вот почему во FreeBSD такие MIB доступны только для чтения. Попытки изменить их ничего не нарушат, однако будет выдано предупреждение о том, что сделать это нельзя.

Но есть MIB, значения которых изменять можно. Рассмотрим следующий MIB:

```
vfs.usermount: 0
```

Его значение определяет, могут ли пользователи монтировать такие носители как CD или дискету. По умолчанию оно равно 0, или off (возможность «выключена»). Для разрешения такого монтирования укажите ключ `-w` команды `sysctl` (чтобы присвоить этому MIB значение 1).[1]

```
# sysctl -w vfs.usermount=1
vfs.usermount: 0 -> 1
#
```

Sysctl возвращает аккуратное маленькое сообщение, в котором показаны предыдущее и новое значения. Это все, что надо сделать для изменения sysctl.

Установка sysctl при начальной загрузке

Параметры настройки ядра, значения которых надо установить во время начальной загрузки, должны быть введены в */etc/sysctl.conf*. В этом файле необходимо указать все устанавливаемые sysctls и их требуемые значения.

Например, для того чтобы пользователи могли монтировать файловые системы, надо настроить параметр `vfs.usermount`. Для этого в *sysctl.conf* необходимо добавить отдельную строку:

```
vfs.usermount=1
```

Конфигурирование ядра с помощью loader.conf

Некоторые этапы конфигурирования ядра должны предшествовать загрузке системы. Так, во время первоначального опроса ядром жесткого диска IDE драйвер устройства определяет, кэшировать ли операции записи. Это решение должно приниматься при обнаружении устройства, на этапе начальной загрузки, после чего решение изменить нельзя. Точно так же, если в системе есть сетевая карта, то модуль ядра с соответствующим драйвером надо загружать до начальной загрузки. Именно здесь пригодится системный загрузчик (system loader).

Загрузчик выполняет много функций. Он находит жесткий диск, на котором находится ядро, загружает ядро в память, инициирует процесс загрузки и поставляет информацию ядру. Самая важная часть информации, передаваемой ядру, – это параметры настройки ядра (sysctl MIB), которые необходимо установить при начальной загрузке.

Самый распространенный способ настройки системного загрузчика – это редактирование конфигурационного файла. Кроме того, команды задания конфигурации можно вводить вручную в приглашении загрузчика `ok>`. Долговременные изменения лучше прописывать в */boot/loader.conf*.

---

[1] В самых последних версиях FreeBSD этот ключ необязателен; просто задайте выражение присваивания.

Есть два важных файла *loader.conf*: */boot/loader.conf* и */boot/defaults/loader.conf*. Второй файл рассмотрен в главе 8, а здесь речь пойдет только о */boot/loader.conf*. Записи в */boot/defaults/loader.conf* представляют собой значения, устанавливаемые системой по умолчанию, и все изменения в */boot/loader.conf* перекрывают эти значения.

К двум основным функциям *loader.conf* относятся загрузка модулей ядра и выдача указаний для драйверов устройств. Такие указания представляют собой перечень парметров настройки, которые можно установить только при начальной загрузке.

В */boot/defaults/loader.conf* содержится много параметров, которые могут быть полезны при различных обстоятельствах. Например, с их помощью можно выбирать загрузку другого ядра, отличного от заданного по умолчанию, или предписать выдачу подробной информации при загрузке. В качестве справки ниже представлен фрагмент */boot/defaults/loader.conf*.

```
kernel="/kernel"
kernel_options=""

userconfig_script_load="NO"
userconfig_script_name="/boot/kernel.conf"
userconfig_script_type="userconfig_script"
. . .
```

Для изменения параметра, заданного по умолчанию, надо скопировать соответствующую строку из этого файла в */boot/loader.conf* и внести в нее необходимые изменения.

Например, первая запись в предыдущем листинге представляет собой имя файла ядра, уже встретившееся нам в примере с sysctl. Предположим, что при работе на удаленной машине необходимо в следующий раз загрузить другое ядро, не копируя его в */kernel*. Для этого достаточно изменить первую строку. Это sysctl, который, очевидно, устанавливается только при начальной загрузке.

Рассмотрим два специфичных примера: выдачу указаний драйверам устройств и автоматическую загрузку модулей ядра.

Выдача указаний драйверам устройств

Первая функция *loader.conf* состоит в выдаче указаний драйверам устройств. (Если драйвер устройства может выполнять эти указания, они описаны на соответствующей странице документации.)

Мы уже говорили, что драйвер жесткого диска IDE должен знать о необходимости кэширования операций записи еще до загрузки системы. (См. страницу документации ata(4); кэширование операций записи обсуждается в главе 16.) Для включения кэширования записи следует установить флаг `hw.ata.wc` в 1, введя в *loader.conf* следующее:

```
hw.ata.wc="1"
```

Вот и все.

Этот флаг может показаться знакомым; он подозрительно напоминает sysctl MIB. После загрузки системы можно проверить, действительно ли это sysctl:

```
# sysctl hw.ata.wc
hw.ata.wc: 1
#
```

Так оно и есть!

На работающей системе значение этого sysctl изменить нельзя. (Можете попробовать – вреда не будет.) Значение sysctl, предназначенного только для записи, было изменено при загрузке. Хотя таким образом не удастся перейти от старого Pentium к Alpha, такая возможность дает дополнительную гибкость.

Автоматическая загрузка модулей ядра

Как уже говорилось, модули ядра – это его компоненты, которые можно запускать (или загружать) и выгружать при необходимости. Такая возможность позволяет экономить системную память и улучшает гибкость системы.

Автоматическая загрузка модулей ядра довольно проста и по умолчанию в *loader.conf* представлено много примеров. Для автоматической загрузки модуля надо воспроизвести его имя в *loader.conf*, удалить замыкающие символы «.ko» и добавить строку `string_load="YES"`. Например, чтобы автоматически загружать */module/procfs.ko* при начальной загрузке, в *loader.conf* надо добавить следующее:

```
procfs_load="YES"
```

Конечно, выяснить, какой модуль загружать, непросто. Проще всего дело обстоит с драйверами устройств; при добавлении новой сетевой карты или адаптера SCSI есть возможность загружать модуль в ходе начальной загрузки, а не пересобирать ядро. При устранении неполадок в системе модули обычно загружаются исходя из соответствующей документации или чьего-либо совета. Умение правильно выбирать модули приходит с опытом, чтением документации и пониманием задач, стоящих перед системой. Позднее будут представлены конкретные указания для некоторых модулей ядра.

Конфигурирование загрузчика вручную

Если перезагрузки при экспериментировании с модулями и sysctls происходят постоянно, то редактирование */boot/loader.conf* отнимает слишком много времени. Но загрузчик можно настраивать и вручную, во время загрузки. Затем, когда будет найдена подходящая конфигурация, ее можно сохранить в */boot/loader.conf*.

Как уже говорилось, в начале загрузки система FreeBSD запускает десятисекундный обратный отсчет времени. Если нажать любую клавишу и прервать обратный отсчет, появится приглашение для ввода команд загрузчика. В этой командной строке можно устанавливать параметры системы.

О готовности системы к вводу команд можно узнать по появлению приглашения загрузчика:

```
ok
```

Загрузчик – это не UNIX. На самом деле загрузчик представляет собой командную оболочку, написанную на языке Forth.[1] Некоторые команды загрузчика похожи на эквивалентные команды UNIX, но это сделано больше для удобства и не является следствием функционального подобия.

Команды загрузчика

Если в приглашении загрузчика ввести знак вопроса (?), на экране появится краткая справка. Вот несколько наиболее полезных команд:

ls

Команда `ls` выдает список файлов, подобно `ls` в UNIX. По умолчанию отображается корневой каталог, а файлы в каком-либо другом каталоге можно увидеть, указав полный путь к нему.

unload

Команда `unload` освобождает системную память, в которую загружаются ядро и модули, указанные в *loader.conf*.

load

Команда `load` копирует файл в память. Эта команда применяется для добавления модулей ядра и даже для загрузки нового ядра. (Однако новое ядро нельзя загрузить, пока старое находится в памяти (выгружается командой `unload`).)

Например, загрузим драйвер сетевой карты Intel EtherExpress:

```
ok load /modules/if_fxp.ko
ok
```

set

Команда `set` позволяет установить значение параметра. Например, для тестирования кэширования записи IDE нужно присвоить параметру `hw.ata.wc` значение 1.

```
ok set hw.ata.wc=1
ok
```

---

[1] Forth – один из немногих языков программирования, которому достаточно крошечного пространства загрузочной записи компьютера (boot record). Подобным программам, написанным на C, необходимо намного большее пространство. Время от времени находятся добровольцы, желающие переписать загрузчик операционной системы на C, BASIC или на другом языке. После этого о них ничего не слышно.

Загрузка и выгрузка модулей в многопользовательском режиме

Некоторые модули ядра не надо загружать в ходе начальной загрузки; их можно загрузить и выгрузить во время работы системы. Рассмотрим, как выяснить, какие модули есть в системе и как их можно загружать и выгружать.

Просмотр загруженных модулей

Когда загрузка системы завершена, с помощью kldstat(8) можно увидеть список загруженных модулей:

```
# kldstat
Id Refs Address    Size     Name
 1    5 0xc0100000 2d505c   kernel
 2    1 0xc0c6c000 13000    linux.ko
#
```

Из этого листинга видно, что в системе загружены два модуля: ядро (kernel) и модуль для совместимости с Linux (linux.ko, см. главу 10). Каждый модуль содержит подмодули, которые можно увидеть с помощью команды `kldstat -v`, однако надо быть готовыми к тому, что вывод растянется на несколько сотен строк.

Загрузка и выгрузка модулей

Загружать и выгружать модули следует посредством команд `kldload(8)` и `kldunload(8)`. Например, для загрузки стандартной заставки консоли («звездное небо») введите такую команду:

```
# kldload /modules/warp_saver.ko
#
```

Выгрузить этот модуль можно следующей командой:

```
# kldunload warp_saver.ko
#
```

Если бы все возможные функции были размещены в ядре, оно было бы намного больше. Таким образом, ядро можно уменьшить, сделав его более эффективным, а модули загружать только по мере необходимости.

Настройка ядра

В какой-то момент становится ясно, что существующее ядро уже не удается подгонять под стоящие задачи только с помощью модулей и sysctl. Единственное решение в этом случае – собрать собственное ядро. Не стоит волноваться – этот процесс очень прост, если выполнять его последовательно, шаг за шагом.

Ядро, поставляемое по умолчанию, называется GENERIC. Это ядро предназначено для работы с широким спектром аппаратных средств, но оно не обязательно будет работать *хорошо* или оптимально. GENERIC прекрасно запускается на системах i486 и более поздних. Однако GENERIC, ориентированное на «наименьший общий знаменатель», не использует преимущества более современных систем x86, обладающих более развитыми функциями и оптимизированных, благодаря чему они более производительны.

Сборка собственного ядра позволит увеличить производительность, внести дополнительную функциональность или поддержку новых аппаратных средств.

Подготовка

Для сборки ядра необходим его исходный код.Если читатель последовал совету, данному в главе 1, то все уже готово. Если нет, надо вернуться в программу установки и загрузить исходный код ядра или перейти к главе 6 и использовать CVSup.

Если неизвестно, инсталлирован ли исходный код ядра, поищите каталог */sys*. Если он существует и в нем имеются файлы и подкаталоги, значит, искомый код у вас есть.

Перед созданием нового ядра выясните, какие аппаратные средства присутствуют в системе. Порой это нелегко, поскольку торговая марка на компоненте необязательно имеет хоть какое-то отношение к возможностям и отличительным чертам устройства. В конце концов, сетевые карты, совместимые с NE2000, производят многие компании. Даже если на коробке написано «3com», микросхемы чипа могут сообщать о себе как об устройствах «ne2-000».[1] Примерно так компании, подобные Linksys[2], занимаются ребрэндингом недорогих сетевых карт, имеющих совершенно разную начинку. На всех коробках написано «Linksys», но чипы говорят о другом, – в зависимости от месяца изготовления.

К счастью, машины на базе PCI имеют усовершенствованную систему распознавания аппаратных средств, и FreeBSD почти всегда находит эти устройства. Обладателям старых компьютеров с шиной ISA придется обратиться к документации по адаптерам, чтобы выяснить их тип и порядок установки прерываний и портов ввода/вывода.

Чтобы узнать, какие аппаратные средства обнаружила система FreeBSD, лучше всего обратиться к */var/run/dmesg.boot*. В этом файле представлено содержимое буфера с сообщениями ядра, которые накапливались во время загрузки. Другими словами, это тот «мусор», который можно было увидеть при начальной загрузке. (В главе 3 есть пример *dmesg.boot*.) Новичкам сле-

---

[1] На самом деле, размещать метку на чипе намного удобнее, поэтому производители аппаратных средств обычно так и поступают.

[2] Летом 2003 г. куплена компанией Cisco Systems. – *Примеч. ред.*

дует уделить этому файлу немного внимания. Разве можно было себе представить, как много всего есть в компьютере!

В начале строк файла *dmesg.boot* можно найти имена устройств. Каждому аппаратному средству соответствует отдельное имя устройства, обычно состоящее из нескольких букв и номера устройства, например npx0. Буквы обозначают название драйвера (npx). Все устройства пронумерованы начиная с 0.[1] Одно устройство может занимать несколько строк, а идентичные устройства будут иметь последовательные номера.

Ядро восстановления

Из-за плохого ядра система может перестать загружаться, поэтому абсолютно необходимо всегда иметь под рукой хорошее[2] ядро. В процессе установки нового ядра старое сохраняется, но его запросто можно перезаписать.

Если хорошего, надежного ядра нет, события могут развиваться так. Вы забываете предусмотреть активизацию сетевого драйвера в текущем ядре и решаете его пересобрать. Заново собранное ядро становится текущим, а предыдущее (с изъяном) в свою очередь заменит старое, рабочее ядро, которое исчезнет в стране Оз.[3] Когда обнаружится, что новое ядро способно проработать не больше нескольких часов, останется только сожалеть о том, что надежное ядро утеряно.

Обычное место для хранения хорошо себя зарекомендовавшего ядра – */kernel.good*. Перед настройкой ядра сохраните надежное, работающее ядро в */kernel.good*:

```
# mkdir modules.good
# cp kernel kernel.good
# cp -R modules/* modules.good/
#
```

Примечание

Не бойтесь хранить несколько ядер. Некоторые администраторы даже размещают ядра в каталогах, названных исходя из даты ядра. Таким образом, у них есть архив ядер за долгое время. Ядер слишком много лишь в том случае, если они совершенно заполнили корневой раздел.

---

1 Устройства каждого типа нумеруются независимо, например два дисковода будут называться fd0 и fd1, а жесткие диски в этой же системе будут нумероваться как sd0, sd1 и т. д. – *Примеч. науч. ред.*

2 «Хорошее» в данном случае означает «работоспособное». Имеет смысл сделать резервную копию ядра сразу после установки системы и в дальнейшем использовать его как гарантированно работающее ядро. – *Примеч. науч. ред.*

3 Страна Оз (Land of Oz) – сказочная страна из книги Лаймена Фрэнка Баума (1856–1919) «The Wonderful Wizard of Oz» (Удивительный волшебник из страны Оз). Издавались многочисленные вариации на темы Баума, в том числе «Волшебник Изумрудного города» Александра Волкова. – *Примеч. перев.*

Редактирование файлов ядра

Итак, рабочее ядро сохранено и все готово к созданию нового ядра. Для начала обратимся к каталогу */sys/i386/conf*, в котором должны находиться несколько файлов. Самые важные из них для наших целей – GENERIC и LINT. GENERIC – это конфигурационный файл для стандартного ядра, используемый при установке системы. LINT содержит все опции ядра и документацию по ним, включая множество действительно странных опций.

Файлы в */sys/i386/conf* не следует редактировать напрямую. Надо скопировать GENERIC в новый файл и редактировать копию, назвав ее по имени машины (так поступают чаще всего). Например, если сервер называется «webserver», можно сделать так:

```
# cp GENERIC WEBSERVER
```

Теперь откроем новый файл конфигурации в любимом текстовом редакторе. Вот фрагмент файла GENERIC, охватывающий накопители IDE (ATAPI):

```
# ATA and ATAPI devices
device          ata0    at isa? port IO_WD1 irq 14
device          ata1    at isa? port IO_WD2 irq 15
device          ata
device          atadisk            # ATA disk drives
device          atapicd            # ATAPI CDROM drives
device          atapifd            # ATAPI floppy drives
device          atapist            # ATAPI tape drives
options        ATA_STATIC_ID      #Static device numbering
```

Каждая строка в конфигурационном файле ядра представляет собой либо комментарий, либо описание функции ядра. Со знака «решетки» (#) начинаются комментарии, они игнорируются компьютером и нужны лишь для удобства.

В некоторых строках комментарии начинаются с середины строки, описывая содержимое первой части строки. Строки, начинающиеся с «device», описывают драйверы устройств; в приведенном примере есть записи для устройств IDE: для жестких дисков, приводов CD-ROM, накопителей на гибких дисках и накопителей на лентах. Есть также записи для шины IDE на системной плате и для двух ее разъемов.

Другие строки отражают программные свойства, или «опции». В приведенном примере свойство ATA_STATIC_ID включает «нумерацию статических устройств»; об этом будет рассказано в главе 16. Также в файле можно увидеть несколько ключевых слов специального назначения, например «pseudo-device» и «cpu», которые либо являются программными опциями, либо описывают аппаратные средства.

Поскольку ядро GENERIC призвано поддерживать все более или менее распространенное оборудование, в него включено огромное множество сетевых драйверов, драйверов жестких дисков, модулей управления и других функций. Общее правило при создании собственного ядра таково: закомментиро-

вать ненужные записи, чтобы сжать и упростить ядро. Конечно, если ядро модернизируется таким способом, его необходимо пересобирать при изменении аппаратных средств. Если такие замены проводятся постоянно, то, возможно, не стоит копаться в ядре. Но если в системе присутствует специфическое оборудование, а создаваемое ядро будет широко распространяться, необходимо убрать все лишнее.

Создаваемый конфигурационный файл (в данном случае WEBSERVER) начинается с закомментированных строк, описывающих его назначение и содержащих ссылки на официальную документацию FreeBSD. После их удаления новый конфигурационный файл будет начинаться так:

```
machine         i386
cpu             I486_CPU
cpu             I586_CPU
cpu             I686_CPU
ident           GENERIC
maxusers        0
```

machine

Ключевое слово *machine* в предыдущем листинге описывает архитектуру системы. Эту строку не стоит менять, если речь не идет о создании ядра для Alpha на x86.

cpu

Операторы *cpu* описывают предполагаемые функции чипа, которые ядро может поддерживать. Это важно, поскольку новые процессоры применяют команды, которые не реализованы в других процессорах. (Например, сравните Pentium и Pentium MMX.)

В файл конфигурации надо включить только тот процессор, который есть в системе. Тип процессора указан в файле *dmesg.boot*. На моем лэптопе в *dmesg.boot* есть следующие строки:

```
CPU: Pentium III/Pentium III Xeon/Celeron (497.56-MHz 686-class CPU)
  Origin = "GenuineIntel"  Id = 0x681  Stepping = 1

Features=0x383f9ff<FPU,VME,DE,PSE,TSC,MSR,PAE,MCE,CX8,SEP,MTRR,PGE,MCA,CMOV,PAT,PSE
36,MMX,FXSR,SSE>
```

Важная часть в этом описании – выражение «686-class CPU» в конце первой строки. Это означает, что для уменьшения размера ядра и увеличения быстродействия можно удалить строки с I486_CPU и I586_CPU. В результате ядро будет оптимизировано для процессора класса 686, а медленный код для поддержки базового процессора будет удален.[1]

---

[1] И если этот компьютер прикажет долго жить, то после перестановки исправного жесткого диска с системой в компьютер с другим типом процессора система запускаться перестанет. Оптимизация – это всегда выбор между скоростью и универсальностью. – *Примеч. науч. ред.*

ident

Оператор *ident* содержит имя ядра, которое обычно совпадает с именем сервера. Если создаваемое ядро будет устанавливаться на многих машинах, ядру можно присвоить имя, отражающее назначение машины, например WEBSERVER.

maxusers

Значение параметра *maxusers* служит для вычисления размера различных таблиц ядра (а не для задания максимального количества пользователей). Внутренние таблицы ядра, к примеру, определяют возможное количество сетевых соединений и количество одновременно открытых файлов.

Начиная с FreeBSD версии 4.5 ядро анализирует ресурсы системы и присваивает параметру `maxusers` значение, которое, по его мнению, подходит для большинства пользователей. Выражение `maxusers 0` означает, что ядро принимает значения по умолчанию. Такие значения полностью подходят для большинства систем. (Тем не менее при желании значение `maxusers` можно жестко прописать; о том, как это сделать, рассказано ниже.)

Во FreeBSD версии 4.4 и в предшествующих жестко задавать значение maxusers необходимо. На моем лэптопе с X Window значение `maxusers` равно 16. Этого хватает, поскольку я – единственный пользователь системы; независимо от того, сколько причудливых штуковин на рабочем столе я запускаю или сколько веб-страниц просматриваю, я не могу открыть больше файлов или установить больше сетевых соединений, чем позволено при значении `maxusers`, равном 16. Однако на загруженном сетевом сервере это значение можно поднять до 256; этого достаточно, чтобы сервер был готов к обслуживанию тысяч сетевых соединений и открытых файлов.

Если значение `maxusers` слишком мало, система не будет справляться с обслуживанием всех открытых файлов или установленных сетевых соединений. Ядро уведомит о том, что оно не может обслуживать все эти запросы, и будет протоколировать ошибки. Сообщения будут сыпаться на консоль и записываться в */var/log/messages*, явно намекая на необходимость увеличить значение `maxusers`.

Однако не надо присваивать `maxusers` значение выше 256, если в одном разделе не хранится ненормальное количество файлов (например, миллионы) или система не обслуживает множество линий T1.

Основные параметры

После значения `maxusers` в конфигурационном файле представлен ряд базовых параметров, например `INET` для поддержки TCP/IP и `FFS` для поддержки файловой системы UNIX. Также можно обнаружить редко используемые параметры, которые можно удалить. Здесь не обсуждаются все параметры ядра, а представлены только наиболее общие из них и специфичные примеры для их различных типов. Особым образом будут отмечены те, что имеют отношение к серверу Интернета.

Рассмотрим следующие «опции»:

```
options         MATH_EMULATE
```

Старые процессоры (особенно 386 и 486SSX) не имели математического сопроцессора. Если в системе нет математического сопроцессора, параметр MATH_EMULATE необходимо оставить, чтобы сопроцессор эмулировался программно. Однако все современные процессоры имеют математический сопроцессор, и если в системе такой установлен, то данный параметр можно убрать.

```
options         INET
```

Параметр INET обеспечивает поддержку сетевых протоколов, таких как TCP/IP. Его надо оставить.

```
options         INET6                   #Коммуникационные протоколы IPv6
```

Если в сети используется IPv6, INET6 надо оставить. Если нет, то убрать.

```
options         FFS
```

Параметр FFS устанавливает поддержку UNIX Fast Filesystem (быстрая файловая система). Это файловая система FreeBSD по умолчанию. Параметр надо оставить.

```
options         SOFTUPDATES
```

Параметр softupdates – это метод обеспечения целостности диска с помощью FFS. (Опция softupdates довольно подробно рассмотрена в главе 13.) Эту строку следует оставить, если нет особых аргументов против использования softupdates.

```
options         MD_ROOT
```

Если на MFS (Macintosh File System) создаются бездисковые рабочие станции, параметр MD_ROOT нужен. В противном случае его надо убрать.

```
options         NFS
options         NFS_ROOT
```

Эти параметры поддерживают сетевую файловую систему (Network File System, NFS). NFS_ROOT позволяет загружаться с устройства NFS, что редко применяется в серверах Интернета. Если NFS не используется, можно удалить оба параметра.

```
options         MSDOSFS
```

Параметр MSDOSFS поддерживает файловые системы и дискеты формата MS-DOS. Если необходимо монтировать или размонтировать дискеты MS-DOS или на жестком диске также есть операционная система Microsoft, то данный параметр пригодится. Кроме того, эту функциональность можно получить, загрузив модуль msdos.ko.

```
options         CD9660
```

Параметр CD9660 поддерживает стандартную файловую систему CD-ROM. Как и в случае с предыдущим параметром, эту функциональность можно получить, загрузив модуль cd9660.ko.

```
options         PROCFS
options         COMPAT_43
```

Если удалить эти две строки, работа системы будет нарушена. Многие пользовательские программы опираются на функции BSD4.3. Параметр COMPAT_43 обеспечивает совместимость ядра с BSD4.3. Точно так же программы мониторинга процессов опираются на файловую систему процессов (PROCFS).

```
options         SCSI_DELAY=15000
```

Параметр SCSI_DELAY задает количество миллисекунд, в течение которых после обнаружения контроллеров SCSI система FreeBSD ожидает перед опрашиванием устройств SCSI, давая им время «раскрутиться». Если в системе нет устройств SCSI, данную строку надо удалить. Если установлены новые аппаратные средства SCSI, то паузу можно уменьшить до 5000 (5 секунд) и менее.

```
options         UCONSOLE
```

Некоторые программы дают пользователям возможность обращаться к системной консоли при работе с терминалом X Windows. Эта возможность поддерживается ядром с помощью параметра UCONSOLE. Эту строку можно удалить, если не предполагается работа в X либо если эта система не установлена в качестве рабочего стола.

```
options         USERCONFIG
options         VISUAL_USERCONFIG
```

Две строки USERCONFIG позволяют включать и выключать устройства перед загрузкой ядра. Сейчас они могут показаться абсолютно ненужными, однако встретив впоследствии в документации FreeBSD рекомендацию «установите это в userconfig», вы пожалеете об удалении этих строк.

```
options         KTRACE
```

Параметр KTRACE включает мониторинг на уровне ядра. Убирать его следует лишь в том случае, если для этого есть совершенно недвусмысленные основания.

```
options         SYSVSHM
options         SYSVMSG
options         SYSVSEM
```

Предыдущие три строки поддерживают межпроцессное взаимодействие в стиле System V. На эту возможность опирается много приложений. Также ее можно реализовать загрузкой соответствующих модулей.

```
options         P1003_1B
options         _KPOSIX_PRIORITY_SCHEDULING
```

Эти две строки обеспечивают поддержку ядром функций POSIX. Многие программы рассчитаны на то, что функции POSIX реализованы в ядре.

Более одного процессора

Если в системе установлено несколько процессоров, необходимы следующие параметры ядра:

```
options         SMP                     # Symmetric MultiProcessor Kernel
options         APIC_IO                 # Symmetric (APIC) I/O
```

Параметр SMP предписывает ядру использовать соответствующий код для поддержки нескольких процессоров; APIC_IO обеспечивает поддержку ввода/вывода для ядер SMP.

При создании ядра SMP надо удалить параметры I386_CPU и I486_CPU из файла конфигурации. FreeBSD поддерживает SMP только в системах, соответствующих спецификации Intel SMP, которая не предусматривает SMP на чипах 386 и 486.

Если в системе один процессор, оставьте эти параметры закомментированными!

Записи устройств

Вслед за параметрами в файле конфигурации расположены записи устройств, которые сгруппированы вполне очевидным образом.

Записи шин

Первые записи устройств – это записи шин, например устройство pci и устройство isa. Убирать их следует лишь в том случае, если в системе действительно нет таких шин. (Количество систем «без поддержки традиционных устройств» (legacy-free), в которых шина ISA скрыта где-то внутри, может оказаться неожиданным; например, в моем совершенно новом лэптопе скрывается старая добрая ISA.) Но в современных компьютерах устройство EISA, вероятно, можно удалить.

Интерфейсы

Далее следуют интерфейсы и устройства IDE/ATAPI (примеры таких устройств были приведены в начале раздела «Редактирование файлов ядра»). Даже если в системе нет устройств IDE, строку «device ata» лучше оставить – в особенности потому, что большинство системных плат имеют один или два контроллера IDE. Параметры для устройств IDE, которых нет в системе, можно удалить.

Следующие записи – контроллеры и карты SCSI, использующие особенности SCSI, в том числе те из них, которые необходимы для приводов ZIP, подсоединяемых к параллельному порту, и устройств хранения USB. Если таких устройств нет, весь раздел можно убрать. Если интерфейс SCSI задействован, удалите записи для контроллеров, которых нет в системе.

```
# SCSI Controllers
device          ahb             # EISA AHA1742 family
device          ahc             # AHA2940 and onboard AIC7xxx devices
. . .
```

За разделом SCSI следуют несколько строк, описывающих драйверы для таких вполне обычных устройств, как клавиатуры, мониторы, порт PS/2 и т. д. Не удаляйте их.

Далее идет достаточно большой список сетевых карт; этот раздел очень похож на разделы SCSI и IDE. Если в ближайшем будущем не предполагается замена сетевой карты, удалите драйверы устройств, которых нет в системе. Разумеется, если в системе нет разъемов ISA, все драйверы ISA можно удалить.

Псевдоустройства

В нижней части файла настройки ядра GENERIC представлен список псевдоустройств. Как подсказывает их название, они всецело создаются программными методами. Например, при удаленном подключении к системе по telnet или SSH (см. главу 13) система должна отслеживать терминальную сессию, посылать символы на удаленный терминал и считывать набранные символы. Система должна воспринимать удаленное соединение точно так же, как физический монитор и клавиатуру, подсоединенные к системе. Для этого она использует псевдоустройство, называемое псевдотерминалом. Поскольку ядро интерпретирует такие терминалы как обычные устройства, они называются псевдоустройствами.

Вот одно из них, для примера:

```
pseudo-device          loop
```

Это интерфейс обратной петли, lo0, сетевой интерфейс, указывающий на саму локальную машину. Если удалить эту строку, многие программы поведут себя интересным образом. Они могут быть очень умными, но такие эксперименты не надо ставить на рабочей системе.

```
pseudo-device          ether
```

Псевдоустройство ether обеспечивает поддержку Ethernet. По всей вероятности, эту строку надо оставить.

```
pseudo-device          sl
```

Псевдоустройство sl предназначено для поддержки межсетевого протокола для последовательного канала (Serial Line Internet Protocol, SLIP). Это старый протокол, который был замещен протоколом соединения «точка-точка» (Point-to-Point Protocol, PPP). Скорее всего, эта строка не нужна, – если только провайдер услуг Интернета не настаивает на этом протоколе.

```
pseudo-device          ppp     1
```

Псевдоустройство ppp предназначено для поддержки PPP на уровне ядра. Такое устройство уже не находит применения – его вытеснил код поддерж-

ки PPP, не входящий в ядро (userland). Вероятно, в этой строке нет необходимости.

Если поддержка PPP средствами ядра действительно применяется, то число после «ppp» – это количество создаваемых устройств PPP.

```
pseudo-device          tun
```

Псевдоустройство tun – это логический пакетный канал (logical packet tunnel). Используется различными программами для обмена пакетами с ядром. Такое псевдоустройство необходимо для поддержки PPP средствами, не входящими в ядро (userland PPP, обычные коммутируемые соединения).

```
pseudo-device          pty
```

Псевдоустройства pty – это псевдотерминалы, используемые для соединений telnet и т. п. Эту строку необходимо оставить.

```
pseudo-device          md
```

Псевдоустройство md предназначено для дисков в памяти (memory disk). Если они не используются, строку можно убрать. Для большинства (но не для всех) серверов Интернета диски в памяти – это просто растрачивание оперативной памяти. Однако изредка в серверах специального назначения (таких, как серверы CVS) диски в памяти необходимы.

```
pseudo-device          gif
pseudo-device          faith
pseudo-device          bpf
```

Псевдоустройство bpf – это пакетный фильтр Беркли (Berkley Packet Filter), позволяющий исследовать пакеты в сети. Он применяется для анализа пакетов, а также для клиента и сервера DHCP. Если такие возможности не задействованы, bpf следует отключить.

Устройства USB

Наконец, после псевдоустройств в конфигурационном файле расположены устройства USB, код поддержки которых может динамически загружаться через kldload. Во многих серверах Интернета USB не используется, поэтому все эти строки можно удалить из конфигурационного файла.

Создание ядра

В предыдущих разделах мы рассмотрели настройку ядра. Рекомендую собрать и запустить описанное минимальное ядро, перед тем как добавлять в ядро поддержку других возможностей. Это позволит понять, что действительно необходимо для надлежащей работы ядра и какая требуется дополнительная настройка.

Примечание

Этапы, описанные в данном разделе, следует применять при создании нового ядра, а не при обновлении существующего. Процедура обновления ядра немного отличается (см. главу 6).

Выбрав и изменив параметры ядра, приступим к его созданию. Прежде всего, воспользуемся командой config(8) для компоновки необходимых файлов и проверки синтаксиса конфигурационного файла. Например, для запуска config с параметром MYKERNEL введите следующую команду:

```
# config MYKERNEL[1]
Kernel build directory is ../../compile/MYKERNEL
Don't forget to do a ``make depend''
#
```

Хотя config не может определить хорошую конфигурацию ядра, она способна найти множество ошибок в конфигурации, если они существуют. Если config обнаруживает что-либо неверное, она сообщает об ошибке и останавливается. Так, если в файл конфигурации включена несуществующая опция, config громко пожалуется. (config всегда напоминает о необходимости запустить *make depend*. Эту команду мы пока не обсуждали, но пропуск этого этапа – стандартная, наиболее типичная ошибка, допускаемая при создании ядра.)

Некоторые сообщения об ошибках довольно очевидны. Например, можно случайно удалить поддержку файловой системы UNIX (UNIX File System, UFS), но включить поддержку инициализации UFS. Одна функция не может обойтись без другой, поэтому config точно скажет, в чем состоит ошибка. Другие сообщения могут быть странными и неясными, и их придется изучить в соответствии с рекомендациями, приведенными в главе 2.

Если config не нашла ошибок, она сообщит, в каком каталоге скомпоновано ядро. В приведенном примере это каталог *../../compile/MYKERNEL*. Перейдите в этот каталог и выполните:

```
# make depend && make all install
```

«Make depend» соединяет воедино все составные части ядра, а также модули ядра, обеспечивая наличие всех необходимых компонентов. Вторая команда, «make all install», компилирует ядро на основе исходного кода и связей компонентов друг с другом.

Теперь надо подождать. Процесс создания ядра отнимет несколько часов на машине с процессором i486, работающим на частоте 25 МГц, или несколько минут на машине с двумя процессорами Pentium и частотой 1 ГГц. При этом по экрану будут пробегать загадочные сообщения о ходе компиляции. На этапе установки текущее ядро будет перемещено в */kernel.old*, а новое установлено как */kernel*.

После создания ядра необходимо перегрузить сервер и прочитать сообщения, выдаваемые при начальной загрузке. В верхних строках должен присутствовать каталог, в который помещено скомпилированное ядро (в нижеследующем примере он показан жирным шрифтом):

---

[1] Выше мы редактировали файл конфигурации ядра, который назывался *WEBSERVER*. Если мы хотим создать именно то ядро, конфигурацию которого редактировали там, следует набрать config WEBSERVER. – *Примеч. науч. ред.*

```
Copyright (c) 1992-2001 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD 5.0-CURRENT #0: Sun May 20 16:49:05 EDT 2001
   mwlucas@turtledawn.blackhelicopters.org:/usr/src/sys/compile/MYKERNEL
...
```

Такое сообщение означает, что создание ядра было успешным. В системе установлено новое ядро!

Выявление неполадок при создании ядра

Если собрать ядро не удалось, то для выявления неполадок прежде всего необходимо изучить последние строки, выданные при компиляции, а именно при выполнении команды `make depend && make all install`. По этим строкам можно догадаться о причинах ошибки, но они совершенно непонятны людям, которые не дышат, не питаются и не живут исходным кодом.[1] Вот пример строк, выведенных на консоль при неудачном создании ядра:

```
===> sys/modules/xl
cc -O -pipe  -D_KERNEL -Wall -Wredundant-decls -Wnested-externs -Wstrict-prototy
pes  -Wmissing-prototypes -Wpointer-arith -Winline -Wcast-qual  -fformat-extensi
ons -ansi -DKLD_MODULE -nostdinc -I-  -I. -I@ -I@/../include  -mpreferred-stack-
boundary=2 -c /usr/src/sys/modules/xl/../../pci/if_xl.c
❶ /usr/src/sys/modules/xl/../../pci/if_xl.c:155: syntax error before`<'
cpp: output pipe has been closed
*** Error code 1

Stop in /usr/src/sys/modules/xl.
*** Error code 1

Stop in /usr/src/sys/modules.
*** Error code 1

Stop in /usr/src/sys.
*** Error code 1

Stop in /usr/src.
*** Error code 1

Stop in /usr/src.
*** Error code 1
```

Согласно верхней части этого сообщения, компилятор пытается собрать работающее ядро на основе исходного кода, расположенного в каталоге *sys/modules/xl*. Далее представлены команды, которые компилятор пытается выполнить, – это строки, начинающиеся с `cc -O`. На экране это одна очень длинная строка, заканчивающаяся как раз над символом ❶.

---

[1] Что касается меня, я предпочитаю дышать воздухом, питаться продуктами и жить своей жизнью, однако некоторые люди, похоже, живут своими компьютерами.

В следующей строке (❶) можно увидеть код ошибки (`syntax error before `<'` (синтаксическая ошибка перед `<')), а также номер строки и имя файла. Эта ошибка прервала процесс компиляции, вызвав каскад ошибок. Если нельзя собрать один модуль ядра, то не собрать и модули в целом, а значит, не собрать и само ядро. Поэтому все заканчивается потоком сообщений об ошибках и аварийным завершением компиляции.

К счастью, во FreeBSD перед установкой чего-либо надо полностью скомпилировать все ядро. Благодаря этому система защищена, а результаты неудачной компиляции по-прежнему находятся в каталоге, созданном программой config. Этап, на котором прекращено создание ядра, известен (код, начинающийся с «cc»). Также известна ошибка, проявившаяся на этом этапе (`syntax error before `<'`). Каскад последующих ошибок, по сути, не относится к делу; неудача на одном этапе прервала весь процесс.

Не теряйтесь, если не понимаете этих ошибок, их не понимают многие. Просто обратитесь за помощью (см. главу 2). Лучше всего начать с изучения архива почтовой рассылки FreeBSD-questions. Скопируйте последние строки, выданные при компиляции (`if_xl.c:155: syntax error before`<'`), вставьте их в окно поиска и посмотрите, кто сталкивался с этой проблемой. Если ничего не нашли, поищите следующую строку (`cpp: output pipe has been closed`).

Если в архиве почтовой рассылки ничего не обнаружено, пошлите сообщение по адресу *FreeBSD-questions@FreeBSD.org*. В письмо включите следующую информацию:

- Последние строки, выданные при неудачной компиляции
- Номер версии FreeBSD
- Содержимое */var/run/dmesg/boot*
- Вывод `uname -a`
- Конфигурационный файл ядра

Бывает, что проблема решается довольно просто, и если представить всю эту информацию, то можно рассчитывать на ответ с рекомендациями. Как правило, такие ошибки вызваны неправильной настройкой ядра.

Загрузка с запасного ядра

Итак, что делать, если новое ядро не работает? Возможно, при конфигурировании не указан драйвер устройства либо удалено псевдоустройство ppp и не удается установить коммутируемое соединение с Сетью. Без паники, не все потеряно. Старое ядро сохранено, верно? Хорошо. Вот что следует предпринять.

Прежде всего, прервите загрузку, нажав любую клавишу (см. раздел «Конфигурирование загрузчика вручную» ранее в этой главе). Приглашение на ввод команды загрузчика означает, что ядро уже загружено. Прежде чем загружать другое ядро, необходимо выгрузить текущее ядро и соответствующие модули. Для этого выполните команду:

```
ok unload
ok
```

Теперь ядро выгружено, а текущим является корневой каталог. Посмотреть, какие ядра есть в этом каталоге, можно при помощи команды `ls`.

Затем выберите ядро, загрузите его (load) и выполните начальную загрузку системы (boot). (Убедитесь, что загружаются и все необходимые модули ядра.)

```
ok load /kernel.good
ok load /modules/if_fxp.ko
ok boot
```

Теперь система должна начать загрузку с выбранным ядром.

Примечание

Если хорошее ядро не сохранено, а новое и старое ядра неработоспособны, не отчаивайтесь. При установке FreeBSD ядро GENERIC помещено в */kernel.GENERIC*. С его помощью можно по крайней мере получить приглашение на ввод команд или загрузиться в однопользовательском режиме.

Добавления в ядро

Если все прошло хорошо, в системе есть работающее минимальное ядро. Сейчас к нему можно добавлять функции и выполнять тонкую настройку.

LINT

Список всех функций ядра находится в файле */sys/i386/conf/LINT*. В нем представлены все опции, драйверы, а также часть документации.

Если в системе есть аппаратные средства, поддержка которых не предусмотрена в ядре GENERIC, обратитесь к LINT. Некоторые из функций малопонятны, но они пригодятся при наличии особых аппаратных средств. Например, FreeBSD поддерживает особые возможности процессора IBM BlueLightning, которые позволяют владельцам систем на его основе использовать все возможности процессора.

Рассмотрим типичную запись в файле LINT:

```
# CPU_PPRO2CELERON enables L2 cache of Mendocino Celeron CPUs. This option is useful
# when you use Socket 8 to Socket 370 converter,because most Pentium Pro BIOSs
# do not enable L2 cache of Mendocino Celeron CPUs.
# (Перевод: CPU_PPRO2CELERON активизирует кэш второго уровня процессоров
# Mendocino Celeron. Эта опция полезна при использовании переходника
# «Socket 8 - Socket 370», т.к. в большинстве BIOS Pentium Pro
# кэш второго уровня процессоров Mendocino Celeron не активизирован.)

options         CPU_PPRO2CELERON
```

Здесь говорится о том, что при наличии на системной плате переходника «Socket 8 – Socket 370» параметр CPU_PPRO2CELERON активизирует кэш второго уровня. Поскольку FreeBSD работает и на устаревшем оборудовании, такая настройка аппаратных средств – не редкость. В подобных системах для расширения возможностей применяются дополнительные адаптеры. Хотя такая ситуация не настолько распространена, чтобы служить основанием для отражения в ядре GENERIC, этот параметр там можно найти. Просмотрите файл *LINT*, чтобы выяснить, какие возможности доступны.

Примечание

Если конфигурация ядра LINT включает в себя все возможные параметры, то почему бы их не использовать? Потому что некоторые функции противоречат друг другу. Например, параметр CPU_PPRO2CELERON говорит ядру, что на системной плате, предназначенной для Pentium Pro, работает процессор Celeron. Между тем, параметр CPU_RSTK_EN активизирует стек возврата в процессоре Cyrix 5x86. Процессора Celeron, работающего на плате для Cyrix 5x86, не существует, а если бы он существовал, то, по всей видимости, не использовал бы описанный адаптер.

Исправление ошибок, связанных с параметрами

Ошибки обусловлены применением определенных параметров. Например, у моего знакомого есть несколько веб-серверов, собранных на аппаратных средствах i386. Когда один сервер оказался слишком перегружен, чтобы обслуживать нескольких сотен веб-страниц в секунду, на консоли стали появляться сообщения, подобные этому:

```
Jun  9 16:23:17 ralph/kernel: pmap_collect: collecting pv entries --
suggest increasing PMAP_SHPGPERPROC
```

Если эта ошибка игнорировалась, происходила авария системы. Мой знакомый обратился ко мне за помощью.

Читая описание ошибки в файле протокола и исследуя *LINT*, я нашел следующее:

```
# Set the number of PV entries per process. Increasing this can
# stop panics related to heavy use of shared memory. However, that can
# (combined with large amounts of physical memory) cause panics at
# boot time due the kernel running out of VM space.
# (Перевод: Устанавливает количество записей (pv-entry) таблицы страниц на один
# процесс. Увеличение этого числа может предотвратить неполадки, связанные
# с нехваткой разделяемой памяти. Однако при большой емкости физической памяти это
# может вызвать неполадки во время начальной загрузки из-за выхода за пределы
# пространства виртуальной памяти.)
#
# If you're tweaking this, you might also want to increase the sysctls
# "vm.v_free_min", "vm.v_free_reserved", and "vm.v_free_target".
# (Перевод: При настройке этого параметра надо также увеличить значения:
# sysctls "vm.v_free_min", "vm.v_free_reserved" и "vm.v_free_target".)
#
```

```
# The value below is the one more than the default.
# (Перевод: Нижеприведенное значение превышает значение по умолчанию.)
#
options         PMAP_SHPGPERPROC=201
```

Прочитав это объяснение, мы приступили к решению проблемы. Прежде всего, сохранили старое ядро в */boot/kernel.pmap-crash*. (Это было не самое лучшее ядро, но я хотел иметь его под рукой, если новое окажется еще хуже.) Затем мы подняли PMAP_SHPGPERPROC до 400, а емкость оперативной памяти увеличили до 192 Мбайт. (Да, эта недорогая система обслуживала несколько сотен веб-страниц в секунду, имея в распоряжении 64 Мбайт оперативной памяти, один диск IDE и CPU Celeron 433!) После внесения изменений в config и сборки нового ядра проблема была устранена, а период работоспособного состояния сервера составляет в среднем 30 дней.

Не будь возможности тонкой настройки ядра, единственным выходом стала бы покупка дополнительных аппаратных средств. Конечно, аппаратное обеспечение этой системы характеризуется низкой производительностью. Но если незначительная программная настройка позволяет системе решать поставленные задачи, почему бы это не использовать? Если вам так хочется потратить деньги, пошлите мне чек.

Настройка производительности ядра

Поговорим о настройке производительности.

Самое узкое место ядра – это сетевые mbufs. В главе 5 будет показано, как mbufs задействуют участки памяти, которые ядро использует для обслуживания сетевых соединений. mbufs представляют собой не количество сетевых соединений, которые может обслуживать сервер, а память, необходимую для поддержки сетевых соединений. Одно соединение может использовать несколько mbufs. (Перед настройкой mbufs можно прочитать о них в главе 5; поскольку здесь обсуждается конфигурирование ядра, настройка mbufs будет рассмотрена сейчас.)

Количество mbufs пропорционально значению параметра ядра MAXUSERS, рассмотренного ранее в этой главе, однако на высокопроизводительном сервере это количество лучше увеличить. Хотя для этого можно увеличить MAXUSERS, непосредственная настройка количества mbufs довольно типична.

Параметр NMBCLUSTERS управляет количеством mbufs, создаваемых ядром. (Этого параметра нет в конфигурационном файле *GENERIC* и его надо добавить. NMBCLUSTERS есть в файле *LINT*.)

```
options         NMBCLUSTERS=1024
```

Для кластеров сетевых mbufs выделяется память ядра, поэтому значение NMBCLUSTERS нельзя поднять до миллиона и забыть об этом, так как памяти может не хватить для других задач системы. Ядро должно обслуживать открытые файлы и поддерживать веб-сервер, не правда ли?

Один NMBCLUSTER отнимает около 2 Кбайт памяти, поэтому в предыдущем примере для поддержки сети резервируется 2 Мбайт памяти ($2 \times 1024 = 2048$; 1 Мбайт = 1024 Кбайт). Это небольшой объем для современных компьютеров, но он существенен для 486-х систем, на которых может работать FreeBSD. Вот почему здесь необходима точная настройка.

Необходимое количество кластеров mbufs вычисляется по количеству сетевых соединений, открытых в часы пик. Такие данные может предоставить команда netstat(1). netstat покажет количество сетевых соединений в системе, включая TCP, UDP, обратную петлю и соединения через сокеты UNIX. В случае с mbufs нужно учитывать только TCP и UDP, поэтому вывод netstat можно пропустить через grep(1). Наконец, программа wc(1) для подсчета слов вычислит количество строк в выводе. Это значение соответствует количеству соединений TCP и UDP, обслуживаемых системой в данный момент.[1] Вот эти команды:

```
# netstat -na | grep tcp | wc -l
     427
# netstat -na | grep udp | wc -l
     377
#
```

Примечание

Для того чтобы выяснить, сколько сетевых mbufs используется в тот или иной момент, запустите netstat -m. netstat будет подробно рассмотрен в главе 5.

Как видно из результатов, в системе есть 427 текущих соединений TCP и 377 активных соединений UDP. В итоге примерно 800. Чтобы учесть возможные всплески, нужно удвоить значение, соответствующее количеству соединений при типичной нагрузке.

Теперь, когда известно, сколько соединений надо обслуживать, выясним, сколько памяти необходимо для каждого соединения. Для каждого соединения TCP нужны пересылочный и приемный буферы. Их текущий размер (в битах[2]) можно узнать из sysctls net.inet.tcp.sendspace и net.inet.tcp.recvspace.

```
# sysctl net.inet.tcp.sendspace
net.inet.tcp.sendspace: 16384
```

[1] Вниманию новичков в администрировании UNIX. Помните, во введении шла речь о командах UNIX, представляющих собой отдельный язык? Вот хороший пример. Мы объединили маленькие команды и получили результат без утомительного подсчета и поиска в строках вывода. Может создаться впечатление, что администраторы UNIX чрезвычайно умны. На самом деле многие из них замечательно ленивы.

[2] В оригинале «bits» – опечатка, на самом деле этот размер вычисляется в байтах. Подробнее об использовании sysctl для настройки параметров TCP можно прочесть по адресу *http://proj.sunet.se/E2E/tcptune.html*. – *Примеч. науч. ред.*

```
# sysctl net.inet.tcp.recvspace
net.inet.tcp.recvspace: 16384
```

С байтами трудно работать, поэтому преобразуем эти значения в килобайты; 16 384 / 1024 = 16, значит, каждый буфер в этой системе занимает 16 Кбайт. (Размер буфера по умолчанию изменился между выпусками FreeBSD 4.4 и 4.5, поэтому это значение необходимо проверять на каждой системе!) Поскольку любому сетевому соединению необходимы входной и выходной буферы, для каждого соединения TCP необходимо 32 Кбайт.

Точно так же, буфер необходим для каждого соединения UDP. Механизм обслуживания соединений UDP не подлежит существенной настройке, но здесь будет разумно допустить, что для каждого соединения UDP необходимо столько же памяти, сколько требуется для соединения TCP.

Итак, для каждого соединения нужно 32 Кбайт, а «средний пик» нагрузки составляет 800 соединений. 800×32 Кбайт = 25 600 Кбайт или около 25 Мбайт. (На самом деле 1 Мбайт равен 1024 Кбайт, но здесь такое округление допустимо.) Далее, с учетом возможных пиков и резких скачков, удвоим это значение, что составит 50 Мбайт.

Один кластер mbuf занимает 2 Кбайт, а 1024 кластера – 2 Мбайт. Для mbufs необходимо 50 Мбайт, поэтому умножим 50 Мбайт на 1024 и разделим результат на 2. Получим 25 600 кластеров mbufs. Установим значение параметра NMBCLUSTERS равным 25 600:

```
options         NMBCLUSTERS=25600
```

Примечание

В сетевом сервере значение NMBCLUSTERS следует установить примерно равным четверти емкости физической памяти. 32 Мбайт, выделенные для mbufs, и размер пересылочного и приемного буфера, равный 16 Кбайт, дадут NMBCLUSTERS = 16384. Такое значение может быть недостаточным или слишком большим, но это хорошая начальная величина.

Распространение ядра

При наличии нескольких идентичных серверов не надо вручную собирать ядро на каждом из них; ядро, специально созданное для одного сервера, можно установить на остальных. (В конце концов, ядро – это простой двоичный файл.)

Прежде всего создайте и инсталлируйте одно ядро и всесторонне протестируйте его. Затем заархивируйте */kernel* и */modules* с помощью tar и скопируйте tarball на остальные серверы. Сохраните текущее ядро на каждом из них и разархивируйте tarball, чтобы установить новые */kernel* и */modules*. Перезагрузитесь, и дело сделано.

5

Работа в сети

BSD известна своей сетевой производительностью. В самом деле, сетевой протокол TCP/IP был впервые реализован в BSD. Многие другие операционные системы стали использовать сетевой стек BSD из-за его высокой производительности и свободной лицензии.

Системный администратор должен понимать работу в сети. Большинство администраторов обычно знакомы с основами, но таких, которые понимают взаимодействие сетевых средств, не так уж и много. Знания об адресации IP, понимание работы сетевой маски и симбиотической связи между IP и TCP – вот что отличает новичка от профессионала. Здесь будут рассмотрены некоторые из этих тем.

Если вы знаете о том, что делает сеть /31 по большей части бесполезной, можете пропустить эту главу. В противном случае прочтите ее. Тестирование пройдет позднее – не в этой книге, а в реальном мире.

Сетевые уровни

Сетевые уровни облегчают организацию работы сети. Каждый уровень обслуживает определенную часть сетевого взаимодействия, а информация переходит с одного уровня на другой. Зачастую новички не в полной мере понимают это, но здесь данная тема будет рассмотрена подробно. Важно помнить, что каждый уровень взаимодействует только с двумя соседними уровнями, расположенными выше и ниже.

В классической диаграмме сетевых протоколов ISO представлено семь уровней. Это полностью законченная схема, охватывающая все ситуации. Однако Интернет – это не «любая ситуация», а эта книга не посвящена работе в сети. Поскольку внимание в этой книге уделяется миру Интернета, упростим эту диаграмму и разделим сеть на четыре уровня: приложение, логический протокол, физический протокол и физический уровень.

Примечание

Описания, приводимые в этой главе, неизбежно будут представлять собой обобщения. На эту тему были написаны толстые книги. Моя любимая – книга Стивенса (Stevens) «TCP/IP Illustrated», Addison-Wesley (тома 1–3).[1]

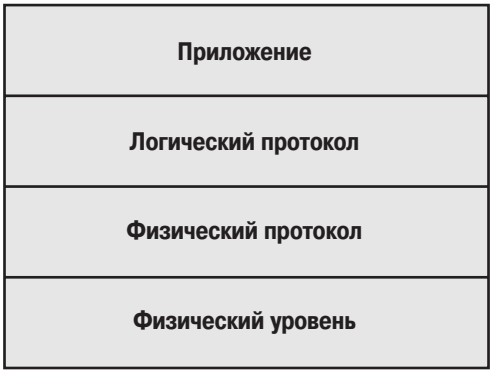

Рис. 5.1. *Четырехуровневая сетевая диаграмма*

Физический уровень

Самый нижний уровень – физический. Он образован сетевой картой и проводом (или другим соединением). Кроме того, этот уровень включает в себя коммутатор (или сетевой хаб) и провод, идущий к маршрутизатору, а также оптоволокно, по которому пакеты передаются от поставщика сетевых услуг через один из коммутационных узлов Интернета (Network Access Point, NAP, пункт доступа к сети) к месту назначения. В этот же уровень входят радиоволны, если применяются беспроводные сети. Без таких каналов связи ничего не работает. Физический протокол всецело опирается на работу этого уровня.

С проводами все просто. Если провод не поврежден и соответствует требованиям, задаваемым физическим протоколом, то дело в шляпе. Если нет, дело табак.

Некоторые физические протоколы были реализованы поверх многих физических уровней. Передачу пакетов Ethernet могут обеспечивать полдюжины различных кабелей. Незначительно изменив драйверы устройств и значительно – сетевые карты, можно изменить физический уровень, не затронув при этом другие уровни. Точно так же, отдельное соединение может вовлекать провода различных типов. Одна из функций маршрутизаторов Интернета состоит в том, чтобы преобразовывать пакеты при переходе с физического уровня одного вида на физический уровень другого вида.

[1] Стивенс У. Р. «Протоколы TCP/IP. В подлиннике», ISBN 5-94157-300-6, БХВ-Санкт-Петербург, 2003 г.

Физический уровень не имеет механизмов принятия решений и развитой логики; единственная его задача – подчиняться физическому протоколу.

Уровень физического протокола

На уровне физического протокола дело становится интереснее. Физический протокол обеспечивает взаимодействие по проводам. Он кодирует пакеты в последовательность нулей и единиц, которые посылаются по физическому носителю соответствующим образом. Например, Ethernet использует адреса протокола управления доступом к среде (Media Access Control, MAC) и протокол разрешения адресов (Address Resolution Protocol, ARP);[1] для коммутируемого соединения используется протокол «точка-точка» (Point-to-Point Protocol, PPP; обычно применяется для установки соединения из дома). Физический протокол должен понимать, как общаться с устройствами по этим адресам. Он отвечает за кодирование и декодирование сообщений для этих устройств.

Ethernet и PPP – это основные физические протоколы, хотя FreeBSD поддерживает и множество других протоколов, таких как асинхронный режим передачи (Asynchronous Transfer Mode, ATM) и цифровая сеть связи с комплексными услугами (Integrated Services Digital Network, ISDN), а также их комбинации, например PPP поверх Ethernet (используется в широкополосных сетях). Каждый из этих протоколов предъявляет специальные требования, и хотя здесь будет более или менее подробно обсуждаться Ethernet, надо понимать, что другие протоколы связи тоже существуют.

Физический протокол обменивается информацией с физическим уровнем и уровнем логического протокола.

Уровень логического протокола

Логические протоколы, такие как межсетевой протокол (Internet Protocol, IP) и протокол управления передачей (Transmission Control Protocol, TCP), обеспечивают IP-адресацию и операции с портами за счет обмена информацией между физическими протоколом и приложением.[2] Одновременно можно использовать несколько логических протоколов.

Есть много логических протоколов. (В файле */etc/protocols* представлен полный список.) Протоколы, о которых в основном пойдет речь, – это уже упомянутые IP и TCP, протокол управляющих сообщений в Интернете (Internet Control Message Protocol, ICMP) и протокол передачи пользовательских дейтаграмм (User Datagram Protocol, UDP).

---

[1] ARP – это протокол из семейства TCP/IP, обеспечивающий преобразование IP-адреса в MAC-адрес (MAC address) для пакетов IP. – *Примеч. перев.*

[2] Это означает, что, с одной стороны, приложение сообщает подпрограмме TCP/IP, какая программа отправляет пакет и какая – ждет ответ, а с другой стороны, подпрограмма TCP/IP умеет общаться с драйверами физических протоколов и обмениваться с ними пакетами. – *Примеч. науч. ред.*

Логические протоколы могут работать бок о бок и даже зависеть друг от друга. В передаваемом пакете есть флаг, который сообщает тип протокола.

Internet Protocol

Internet Protocol (IP) – это связующая основа Интернета; предполагается, что каждое устройство Интернета говорит на языке IP. IP предоставляет основные, базовые функции, такие как сетевая адресация и маршрутизация пакетов, а также фундаментальную инфраструктуру, используемую другими логическими протоколами. Можно обойтись без TCP и других подобных протоколов, но без IP не будет и Сети.

Примечание

В этой книге рассматривается протокол IP версии 4; IP версии 6 появился не так давно. Хотя FreeBSD прекрасно поддерживает IPv6, этот протокол недостаточно распространен, чтобы обсуждать его здесь. Будем надеяться, что ко времени выхода нового издания этой книги IPv6 найдет широкое распространение, поскольку он способен снять много ограничений IPv4 и устранить грубые ухищрения, реализованные для их преодоления.

IP-адреса

IP-адрес – это 32-разрядное число, обычно разделенное на четыре группы по 8 бит в каждой. В переводе на человеческий язык это означает четыре числа, принимающих значения от 0 до 255, разделенных точками. Например, 192.168.1.87 – это действительный IP-адрес, а 192.259.0.87 – нет, поскольку одно из чисел превышает 255. Ну, с «адресом» 607.322.843.999 все ясно.

Каждое устройство в Интернете имеет уникальный IP-адрес, – если не применяется преобразование сетевых адресов (Network Address Translation, NAT) или другие трюки.

Прикладной уровень

Наконец, существует прикладной уровень – приложение, с которым имеет дело пользователь. Это может быть веб-броузер, текстовый редактор или клиент, выполняющий команды оболочки. Приложение заботится только о пользовательском интерфейсе и логическом протоколе. (Конечного пользователя можно считать еще одним уровнем, но трудности, связанные с ним, выходят за рамки темы этой книги.)[1]

[1] Как бы вы ни отчаялись или ни были раздражены, проверка «файловой системы» пользователя или обновление его «BIOS» – это не самая лучшая идея. Уровень пользователя можно реконфигурировать лишь в процессе, который называется «обучение». Не стоит ожидать от него слишком много.

Сеть на практике

Теперь, когда ясен каждый уровень, рассмотрим их подробно. Обсудим, как сетевые уровни работают с точки зрения офисного настольного компьютера, подсоединенного к сети через Ethernet.

Пусть пользователь набирает адрес *www.absolutebsd.com* в своем веб-броузере. Веб-броузер должен знать, как передать запрос на уровень ниже. Для этого он преобразует имя хоста в IP-адрес (ряд чисел, например 192.168.1.84).

Примечание

По умолчанию сервер обслуживает запросы с помощью сетевых портов, или логических идентификаторов. Далее порты будут рассмотрены более подробно, а сейчас необходимо понимать, что каждый сервис, который предоставляет компьютер, запущен на уникальном порту. Веб-серверы обычно запускаются на порту TCP с номером 80. Запрос, посланный броузером по заданному IP-адресу и на заданный порт, переходит на уровень ниже.

Далее, на уровне логического протокола анализируется запрос, поступивший от приложения. Поскольку приложение обратилось с запросом на установление соединения TCP/IP, логический протокол выделяет для этого соответствующие системные ресурсы. Запрос разбивается на части допустимого размера, которые можно посылать по IP. Эти части называются *пакетами*.

С этого момента логический протокол не думает о реальном запросе, полученном от приложения; вместо этого он должен доставить пакеты по указанному адресу. На уровне IP проверяются внутренние таблицы, чтобы выяснить путь до компьютера с требуемым адресом. Затем данные распределяются по пакетам, к ним добавляется информация о маршрутизации на уровне IP и пакеты передаются на уровень физического протокола.

На уровне физического протокола изучается запрос, поступивший с уровня логического протокола (здесь ничего не известно о веб-броузере; все, что требуется, – это доставить каждый пакет до места назначения). На этом уровне проверяется адрес протокола физического уровня (MAC-адрес), заданный как адрес назначения. Кроме того, к каждому пакету добавляется информация Ethernet. Порция данных «пакет плюс физический протокол» называется *сегментом*. Наконец, все это передается с уровня физического протокола на физический уровень, где данные преобразуются в нули и единицы и передаются по проводам. Коммутаторы и маршрутизаторы «повторяют» эти нули и единицы на всем пути сегментов к месту назначения.

На пути сегментов тип физического носителя может меняться. Например, Ethernet может перейти в линию T1 (такое преобразование осуществляет офисный маршрутизатор). Затем T1 присоединяется к оптоволокну, которое протянуто по всей стране (такой переход обеспечивают телефонные компании).

Достигнув места назначения (компьютера на другой стороне соединения), сегмент начинает обратный путь по стеку протоколов.

Физический уровень передает каждый сегмент на уровень физического протокола, который выполняет основную проверку сегментов, чтобы убедиться

в их целостности. После того как станет ясно, что сегмент цел, из него удаляется информация Ethernet и воссоздается пакет, который передается на уровень логического протокола.

Логический протокол, в свою очередь, выполняет собственную проверку. Помните, как логический протокол разбил запрос на пакеты в целях упрощения передачи? Теперь он собирает пакеты в информационный поток и перенаправляет его к приложению. Затем приложение получает ответ и может отобразить веб-страницу.

Конечно, предполагается, что все это происходит очень быстро.

Кажется, что такой процесс подразумевает огромный объем работы, но это замечательный пример *абстракции интерфейса*. Каждый уровень знает лишь самое необходимое об уровнях, находящихся выше и ниже, поэтому при желании можно менять целые уровни. Когда создается новый физический протокол, другим уровням не о чем беспокоиться; логический протокол передает запрос на уровень нового физического протокола и оставляет за ним всю внутреннюю работу. При появлении сетевой карты нового типа надо лишь написать драйвер для физического протокола; приложение и логический протокол здесь роли не играют.

mbufs

BSD оптимизирует сетевую работу с помощью *mbufs*. mbuf – это отдельный участок памяти, выделенный для сетевых операций, который живет в ядре. Пакет появляется на свет как mbuf. Вместо копирования содержимого каждого пакета на уровень ниже каждый уровень ОС перемещает вниз весь mbuf. Копирование одних данных отнимает больше времени и ресурсов, чем передача данных в их текущем обрамлении.

Более того, mbufs разработаны так тщательно, что не требуют изменений при передаче. Когда логический протокол создает mbuf, он оставляет пространство в начале и конце участка памяти, которое предназначается для заголовков физического протокола. За счет этого снижается объем копирования. Пакет становится сегментом в пределах одного mbuf.

Программисты на C распознают здесь указатель. Для операций доступен указатель на mbuf, тогда как сам mbuf остается неизменным. Тем, кто не программировал на C, надо получить только базовые представления о том, что представляет собой mbuf. mbufs упоминаются при изучении любого сетевого стека BSD, поэтому важно получить хоть какие-то знания о них.

Что такое бит?

Сетевому администратору частенько встречаются такие термины, как *32 бита* и *48 бит*. Их смысл необходимо понимать, чтобы можно было распознавать неверные значения.

Мы знаем, что компьютер интерпретирует данные как нули и единицы. Одна единица или один ноль составляют один бит. Когда протокол задает биты,

он указывает это число согласно представлениям компьютера – в двоичной форме. (Двоичные операции, или операции с числами по *основанию 2*, наверняка изучались вами в средней школе и быстро забылись. Пришло время освежить эти знания. Двоичная система – это просто другой способ представления чисел, с которыми люди работают каждый день.)

В десятичной математике (числа с *основанием 10*), ежедневно используемой людьми, задействованы цифры от 0 до 9. Для того чтобы получить число, превышающее самую большую цифру, слева к этой цифре добавляется другая, а текущая становится равной 0. (Это правило «переноса 1», которое вы изучили много лет назад и сейчас используете неосознанно.) В двоичной математике есть только цифры 0 и 1. Чтобы получить число, превышающее самую большую цифру, слева к этой цифре добавляется другая, а текущая становится равной 0. Все то же самое, только цифр меньше.

Вот как десятичные числа преобразуются в двоичные.

Десятичные	Двоичные	Десятичные	Двоичные
0	0	8	1000
1	1	9	1001
2	10	10	1010
3	11	11	1011
4	100	12	1100
5	101	13	1101
6	110	14	1110
7	111	15	1111

Число из 32 бит, например IP-адрес, представляет собой строку из 32 нулей и единиц. Однако это число не представляет собой неделимое целое. IP-адрес разбит на четыре 8-битных числа (причина будет объяснена в следующем разделе.)

Примечание

Преобразование из двоичной формы в десятичную и обратно реализовано во многих калькуляторах. Если такого калькулятора нет под рукой, можно воспользоваться пакетом math/calctool, перенесенным во FreeBSD. Такую операцию может выполнить даже калькулятор Windows, если перевести его в инженерный режим. Возьмите IP-адрес, поместите каждое из четырех составляющих его чисел в калькулятор и преобразуйте их в двоичную форму.[1]

[1] Если этот раздел непонятен, следует почитать что-нибудь о позиционных системах счисления в учебнике математики за 10-й или 11-й класс средней школы, там об этом должно рассказываться. – *Примеч. науч. ред.*

Ethernet

К сети Ethernet может быть подключено много устройств; данные, которые получает та или иная система, необязательно предназначаются именно ей. Системы, подсоединенные через Ethernet, могут напрямую говорить друг с другом. Это одно из преимуществ Ethernet перед другими протоколами, такими как PPP. Однако Ethernet налагает ограничения на физические расстояния между системами, поэтому этот протокол подходит только для офисов, услуг физического хостинга (colocation facilities) и других сравнительно небольших сетей.

На протяжении многих лет для реализации Ethernet применялось много различных физических носителей. Было время, когда большинство кабелей Ethernet представляли собой толстый коаксиальный кабель (coax), а сегодня в большинстве сетей Ethernet задействованы сравнительно тонкие кабели категории 5 (cat5) с восемью тонкими проводами внутри. Ethernet можно реализовать по оптоволокну или, при менее удачном раскладе, по «темному оптоволокну». (*Темное оптоволокно*[1] – это оптоволокно без света. Одно время эта идея казалась хорошей.)

В рамках нашего разговора предположим, что Ethernet в данном случае реализована по кабелю cat5 – сегодня это наиболее распространенный носитель. Так или иначе, теоретические основы для всех физических носителей Ethernet одни и те же.

Широковещание

Ethernet – это широковещательная (broadcast) среда[2] – каждый пакет, посылаемый по сети, доставляется на каждую рабочую станцию. (Далее мы увидим, что сейчас это уже не догма.) Драйвер сетевой платы на той или иной рабочей станции отделяет данные, предназначенные для этой машины, от всех остальных передаваемых пакетов.

Побочный эффект широковещательной сущности Ethernet – возможность «вынюхивать» чужие секреты, собирая все данные, которые посылаются по сети. Хотя такая возможность очень полезна при диагностировании неполадок, она добавляет хлопот с точки зрения безопасности: перехват пароля – тривиальная операция в несовременных[3] сетях Ethernet.

---

[1] «Темным оптоволокном» называют оптическую инфраструктуру (кабели и повторители), готовую к эксплуатации, но не используемую, т. е. проложенную «про запас». Автор называет такой расклад менее удачным, поскольку перед началом эксплуатации потребуется протестировать кабель на предмет работоспособности, возможно, заплатить за его аренду и понести прочие расходы. – *Примеч. науч. ред.*

[2] Однако пакеты не просачиваются сквозь эфир (ether) и не распространяются подобно радиоволнам. Физический провод – обязательное требование.

[3] Современность тут ни при чем. В тех сетях, где все компьютеры соединены с помощью коммутаторов, чужой пакет посмотреть невозможно, а в сетях с концентраторами (хабами) – элементарно. – *Примеч. науч. ред.*

Хотя первые сети Ethernet поддерживали передачу данных со скоростью несколько мегабит в секунду, они переросли исходный замысел и сегодня поддерживают гигабитные скорости. В большинстве современных сетей скорость передачи данных составляет 10/100 мегабит в секунду (Mbps), а сетевые карты Gigabit Ethernet все еще сравнительно дороги. Карты Gigabit Ethernet дешевле $100 только появились на момент написания этих строк.

Разрешение адресов

Каждая сетевая карта Ethernet имеет уникальный идентификатор, или *MAC-адрес*, который используется при взаимодействии с другими хостами. MAC-адрес – это 48-битное число. Когда одна система хочет передать данные другой через Ethernet, она посылает Ethernet-запрос, суть которого можно выразить так: «Какой MAC-адрес отвечает за этот IP-адрес?». Если хост отвечает, отправляемые данные помечаются в соответствии с его MAC-адресом. Этот процесс известен как протокол разрешения адресов (Address Resolution Protocol, ARP).

Узнать текущий MAC и получить данные ARP можно с помощью команды `arp`. Обычно ее запускают как `arp -a`, что позволяет увидеть MAC-адреса и имена всех хостов[1] в сети:

```
# arp -a
? (192.168.1.1) at 0:a0:cc:35:5b:7 [ethernet]
magpire.blackhelicopters.org (192.168.1.222) at 0:4:5a:41:a4:44 [ethernet]
#
```

В этом примере хост *magpire.blackhelicopters.org* имеет IP-адрес 192.168.1.222 и MAC-адрес 0:4:5a:41:a4:44. MAC-адрес – это Ethernet-адрес. Если MAC-адрес «неполный», установить соединение с хостом не удастся. В этом случае необходимо проверить физический уровень (провод) и конфигурацию системы.

Хабы и коммутаторы

Хаб[2] Ethernet – это центральная часть аппаратного обеспечения, обеспечивающего физическое соединение устройств Ethernet. Он просто осуществляет переадресацию и пересылку информации уровня Ethernet между устройствами, которые к нему подключены. Весь поступающий трафик Ethernet хабы переадресуют каждому присоединенному хосту.

Коммутатор предоставляет усовершенствованный способ соединения устройств Ethernet, при котором скорость передачи данных в сети Ethernet улучшается за счет отслеживания MAC-адресов всех подсоединенных уст-

---

[1] Команда `arp -a` показывает текущее состояние таблицы `arp` той системы, где эту команду выполнили. В таблице будут только те хосты, с которыми данная система общалась в последнее время. Если с каким-то хостом соединение не устанавливалось давно, он выбывает из таблицы. – *Примеч. науч. ред.*

[2] Хаб часто называют концентратором, здесь мы предпочитаем название «хаб» для краткости. – *Примеч. науч. ред.*

ройств и перенаправления пакетов тому устройству, для которого они предназначены. Поскольку каждый хост Ethernet имеет ограниченную пропускную способность (например, карта 100 Мбит/с может в секунду управиться с сотней мегабит), коммутация снижает нагрузку на отдельные системы за счет ограничения объема данных, передаваемых каждому устройству. Однако коммутаторы стоят дороже, чем хабы.

Сетевые маски

При подключении компании к Интернету провайдер выделяет ей блок IP-адресов. Эти адреса используются для локальной сети Ethernet. Как правило, этот блок невелик, скажем, 16 или 32 IP-адреса. Если речь идет о небольшой группе идентичных серверов, компания получит лишь несколько IP-адресов. Все зависит от ее потребностей. Размер блока IP-адресов определяет *сетевую маску*.

Раньше довольно часто можно было встретить сетевую маску 255.255.255.0. Также общеизвестно, что неверная сетевая маска приведет к неработоспособности системы. Сейчас такая простая маска встречается все реже и реже. Чтобы понять, в чем тут дело, необходимо разобраться, что представляет собой сетевая маска и как выделяются блоки IP-адресов.

Много лет назад IP-адреса выделялись блоками трех размеров, соответствующих классу A, классу B и классу C. Со временем эта терминология устарела, но мы возьмем ее за основу.

Класс A был очень простым: первое из четырех чисел IP-адресов было фиксированным. InterNIC мог выделить блок IP-адресов класса A, скажем, 10.0.0.0. Оставшиеся три числа владелец IP-адреса мог назначать по своему усмотрению. Например, адреса с 10.1.0.0 по 10.1.1.255 можно было выделить для информационного центра, адреса с 10.1.2.0 по 10.1.7.255 – для бостонского офиса и т. д. Блоки адресов класса A получали только очень большие компании, такие как Ford и Xerox, а также влиятельные академические организации, связанные с компьютерными технологиями.

В блоках класса B фиксированными были первые два из четырех чисел, составляющих IP-адрес. Блок адресов класса B мог выглядеть, например, как 64.29.0.0. Каждый IP-адрес, использовавшийся во внутренней сети, начинался с 64.29, а по своему усмотрению можно было назначать последние два числа. Многие компании среднего размера получили блок адресов класса B.

Подобным образом в блоке класса C фиксированы первые три числа адреса. Обычно такие блоки получали маленькие компании. Провайдер услуг Интернета выделял номер, подобный 209.69.9, и предоставлял владельцу блока возможность назначать последнее число каждого адреса.

Такая схема приводила к растрачиванию IP-адресов. Многие мелкие фирмы не нуждались в 256 IP-адресах, а многие компании среднего размера занимали больше 256, но меньше 65 000 адресов в блоке класса B. И почти никому не требовались все 16 миллионов адресов из блока класса A. Однако таков был выбор, и до бума Интернета такое положение всех устраивало.

Сегодня IP-адреса выделяются вместе с *префиксной длиной* (*prefix length*), отделяемой *косой чертой* (*slash*). Блок IP-адресов может выглядеть так: 192.168.1.128/25. Подобная запись может быть не очень понятной, но делает возможным более тонкое использование классов адресов. Известно, что каждое число в IP-адресе состоит из 8 бит. При использовании классов «фиксированным» считается определенное количество бит – их нельзя изменить в локальной сети. Адрес класса A имеет 8 фиксированных бит, класса B – 16 бит и класса C – 24 бита.

В приведенных примерах IP-адреса того или иного класса не были представлены в двоичной форме, и я не буду заставлять вас выполнять это преобразование. Однако под IP-адресом следует понимать строку из двоичных чисел. В адресах локальной сети можно изменить биты в правой части каждого адреса, но не биты в левой части.

Нет никаких причин, по которым длина каждой части адреса должна быть кратной 8. Префиксная длина – это просто некоторое количество фиксированных битов. Запись /25 означает, что фиксированными являются 25 бит, то есть на один бит больше, чем в адресе класса C. В этом случае изменять можно 7 бит. В следующей записи фиксированные биты представлены единицами, а изменяемые – нулями:

```
11111111.11111111.11111111.10000000
```

Это по-детски просто – конечно, если вы мыслите двоичными категориями. С двоичными числами вовсе не обязательно работать каждый день, но тем, кто не понимает базовые концепции двоичной математики, преобразование чисел из двоичной формы в десятичную кажется совершенно не понятным. В ходе практической деятельности можно научиться видеть десятичные числа в логичной двоичной записи.

Что это означает на практике?

Прежде всего, блоки IP-адресов выделяются в количестве, кратном 2. Если изменяемыми являются 4 бита, то возможны 16 адресов ($2\times2\times2\times2=16$), а если 8 бит, то (2^8) 256 IP-адресов. Если кто-либо говорит, что доступно 13 IP-адресов, то либо речь идет о разделяемой сети Ethernet, либо высказанное утверждение ошибочно.

Сетевая маска – это всего лишь другой способ указать количество фиксированных битов. Двоичное число, состоящее из 8 разрядов, может иметь значение от 0 до 255. В случае префиксной длины /24 сетевая маска будет такой: 255.255.255.0. Если префиксная длина представлена как /25, то фиксированными являются все биты в первых трех числах и один бит в последнем числе. В предыдущем примере последнее число было представлено в двоичной форме так: 10000000. Несложные вычисления по преобразованию двоичного числа в десятичную форму, выполненные на калькуляторе,[1] дадут 255.255.255.128.

---

[1] Такое преобразование стоит выполнить и на бумаге, по крайней мере несколько раз. Таким образом можно научиться гораздо большему. Вперед, смелее!

Нередко можно увидеть IP-адрес хоста вместе с его сетевой маской, например 192.168.3.4/26. Запись /32 представляет собой не сеть, а отдельный хост и применяется, когда необходимо ясно показать, что речь не идет о сети.

Трюки с сетевой маской

Преобразование из десятичной формы в двоичную не всегда доставляет удовольствие. Вот как можно вычислить сетевую маску.

Прежде всего выясним, сколько реальных IP-адресов есть в наличии. Это число будет кратно 2. Почти всегда количество адресов меньше значения, вычисляемого по записи /24 (обычная адресация класса C). Из числа 256 вычтем количество адресов, имеющихся в наличии, и получим последнее число сетевой маски.

Например, при наличии 64 IP-адресов последний компонент сетевой маски вычисляется так: 256 – 64 = 192, а сетевая маска будет иметь вид: 255.255.255.192.

Для того чтобы избежать двоичных преобразований, обратимся к логике. Составление списка IP-адресов, допустимых в той или иной сети, – дело нелегкое. При наличии IP-адреса 192.168.54.187/25 фиксированными являются 25 бит. Значит, используется блок из 128 IP-адресов. Рассмотрим последнее число этого адреса, 187. Конечно, оно находится не в интервале между 0 и 127, а в интервале между 128 и 255. Другие хосты из этого блока имеют IP-адреса от 192.168.54.128 до 192.168.54.255.

Шестнадцатеричные сетевые маски

Все ясно? Хорошо. К сожалению, в UNIX принято представлять сетевую маску в шестнадцатеричной форме (основание 16), а не в десятичной или двоичной. Рано или поздно вы увидите сетевую маску 0xffffff00.

Шестнадцатеричная цифра занимает 4 бита, поэтому каждая 8-битная часть сетевой маски может быть представлена как две шестнадцатеричных цифры. (IP-адреса также можно представить в такой форме, но это не принято.) Шестнадцатеричные числа всегда начинаются с «0x», поэтому распознать их легко.

Для преобразования числа в шестнадцатеричную форму проще всего воспользоваться калькулятором или специальной таблицей. В табл. 5.1 для удобства представлено преобразование «запись с косой чертой/шестнадцатеричная форма/двоичная форма/десятичная форма» для сетевых масок от /24 до /32.

Таблица 5.1. Преобразования сетевых масок

Префикс	Двоичная маска	Десятичная маска	Шестнадцатеричная маска	Доступные IP-адреса
/24	00000000	0	0x00	256
/25	10000000	128	0x80	128

Таблица 5.1 (продолжение)

Префикс	Двоичная маска	Десятичная маска	Шестнадцатеричная маска	Доступные IP-адреса
/26	11000000	192	0xc0	64
/27	11100000	224	0xe0	32
/28	11110000	240	0xf0	16
/29	11111000	248	0xf8	8
/30	11111100	252	0xfc	4
/31	11111110	254	0xfe	2
/32	11111111	255	0xff	1

Неиспользуемые IP-адреса

Теперь ясно, что запись /26 подразумевает 64 IP-адреса. К сожалению, не все из них можно использовать. Первый IP-адрес – это *номер сети* (*network number*). Он предназначен для внутреннего управления сетевыми ресурсами. В любой группе IP-адресов последний номер является широковещательным *адресом* (*broadcast address*). Согласно спецификации IP, каждая машина в сети должна откликаться на запрос, посылаемый по этому адресу. Такая возможность позволяет проверять доступность адресатов и выяснять, какие IP-адреса используются в сети. Например, в типичной сети /24 широковещательный адрес будет иметь вид: x.y.z.255. В конце 90-х годов отправка пакетов по широковещательному адресу применялась при организации атак на сети. Сейчас эта возможность отключена в большинстве операционных систем. В системах BSD поддержку широковещательной адресации можно включить с помощью установки параметра `net.inet.icmp.bmcastecho` в 1.

В любом случае интерфейсу нельзя назначить первый и последний IP-адреса сети. Но можно попробовать.

Если помните, в первой части этой главы я упоминал, что маска /31 практически бесполезна. Такая запись подразумевает два IP-адреса, а первый и последний адреса использовать нельзя. Остается не так уж много места для серверов и даже для клиентов.

Маршрутизация

Итак, в наличии есть IP-адреса для вашей сети Ethernet, а каждый хост локальной сети может найти любой другой хост. Осталось научить системы общаться с другими сетями. Обычно в каждой сети есть свой маршрутизатор или другой внешний шлюз. Такое устройство называется маршрутизатором *по умолчанию* (*default router*). В любой сети маршрутизатор по умолчанию должен быть один и только один. Каждая система, входящая в состав сети, должна знать IP-адрес этого маршрутизатора.

Как только система узнала адрес маршрутизатора по умолчанию, она может «исследовать» (ping) любой ресурс Интернета по его IP-адресу или по имени хоста, если распознаватель (resolver) сконфигурирован правильно (см. описание */etc/resolv.conf* в главе 11).

UDP и TCP

Итак, IP работает, и настало время передавать по нему данные. Для этого программы могут задействовать протокол дейтаграмм пользователя (User Datagram Protocol, UDP). UDP – это, вероятно, самый элементарный протокол, возможный для IP. Он не поддерживает обработку ошибок и проверку данных. В нем не предусмотрены какие-либо средства предотвращения их потери. Несмотря на это протокол UDP может быть хорошим выбором, и его применяют важнейшие сервисы Интернета. Приложение, использующее UDP, зачастую имеет собственные требования к коррекции ошибок, которые не соответствуют возможностям, предоставляемым другими протоколами.

Когда хост передает данные по UDP, он не знает, доставлены ли они до места назначения. И когда хост получает данные по UDP, он не может проверить, откуда пришли эти данные. Хотя пакеты UDP включают в себя адрес отправителя, его легко подделать. По этой причине UDP называют *протоколом без установления соединения* (*connectionless*).

Другой типичный транспорт для передачи данных по IP – это протокол управления передачей (Transmission Control Protocol, TCP). TCP включает в себя средства для исправления ошибок и восстановления пакетов. На каждый отправляемый пакет должно прийти подтверждение от получателя, иначе пакет будет передан повторно. Приложения, использующие TCP, вправе ожидать надежной передачи данных, при условии стабильной работы нижних уровней сети.

В отличие от UDP, TCP – это протокол с установлением соединения. Два хоста должны организовать между собой канал, по которому будут передаваться данные. Этот процесс называется *тройным рукопожатием* (*three-way handshake*). Подробности на данном этапе нас не интересуют, нам достаточно знать, что для организации соединения TCP хостам необходимо проделать определенный объем работы. По завершении обмена данными для разрыва соединения тоже надо выполнить некоторые действия.

Протоколы IP, TCP и UDP можно сравнить с семьей, сидящей за обеденным столом. Родственники передают блюда друг другу. Благодаря протоколу IP известно, где сидит каждый член семьи, а также понятно, что передаче горошка дяде Джиму должна предшествовать передача этого блюда кузине Колин. TCP руководит передачей блюда от дяди Джима кузине Колин и ее ответного «спасибо» – прежде чем дядя Джим займется другим делом. Действия UDP можно уподобить перебрасыванию блинчика тетушке Джейн, – может быть, она его поймает, а может быть, домашний пес перехватит его в воздухе.

Сетевые порты

Вы когда-нибудь замечали, что компьютеры имеют слишком много портов? Что ж, добавим в список порты TCP и UDP. Благодаря наличию сетевых портов один сервер может предоставлять различные сетевые сервисы. Порты являются средством для организации множественных соединений между машинами.

Пакет (TCP или UDP), поступающий в систему, необходимо доставить к определенному порту. Различные порты предоставляют различные сервисы. Например, почтовая служба Интернета называется SMTP. Как указано в файле */etc/services*, SMTP работает через порт 25. Если запрос на соединение TCP запрашивает порт 25, можно сделать вывод, что происходит обращение к почтовому серверу. Порты позволяют организовывать множественные соединения между несколькими машинами.

Файл */etc/services* содержит список номеров портов и сервисов, которые с ними ассоциированы. Почти любой сервис можно запускать на произвольном порту, но таким образом можно сбить с толку другие хосты Интернета, которые пытаются соединиться с вашей системой. Формат файла очень прост: официальное имя сервиса, номер порта, протокол, все псевдонимы (aliases) для этого сервиса и комментарии. Поля разделены табуляцией. Например, одним из первых сервисов на хостах UNIX был сервис qotd (Quote of the Day, цитата дня). В файле */etc/services* можно найти соответствующую запись:

```
qotd            17/tcp    quote        #Quote of the Day
qotd            17/udp    quote        #Quote of the Day
```

Многие сервисы работают через порты TCP и UDP, которым назначены определенные номера, но есть и другие сервисы, работающие только по одному протоколу.

Многие программы читают */etc/services*, чтобы выяснить, какой порт надо использовать (к какому порту *осуществить привязку*, *bind to*). Надо отредактировать этот файл, чтобы в зависимости от программы назначить тот или иной протокол тому или иному порту. В частности, перед началом работы необходимо проверить настройки для управления версиями (см. главу 3). Подобно всем другим стандартам, содержимое файла */etc/services* не является догмой. Служба sshd обычно занимает порт 22, однако я запустил ее на порту 80, чтобы в чрезвычайных обстоятельствах можно было обойти ограничения брандмауэра. Все зависит от программы, задействованной для предоставления того или иного сервиса.

Порты с номерами 1024 и ниже называются *портами с низкими номерами* (*low-numbered ports*). Они зарезервированы для ключевых протоколов инфраструктуры Интернета, таких как DNS, telnet и HTTP. Стандарты их применения словно высечены в камне. Порты с номерами 1024 и выше не настолько стандартизованы, поэтому изредка возникают конфликты, когда различные протоколы пытаются использовать один и тот же порт. По существу, клиент инициирует соединение с портом, номер которого выше 1024, и запрашивает соединение с сервером через порт с низким номером.

Время от времени к */etc/services* обращаются службы, работающие поверх протоколов, отличных от TCP и UDP. Некоторые службы используют протокол доставки дейтаграмм (Datagram Delivery Protocol, DDP). О них беспокоиться не надо – для этого нет оснований. В то же время почти любой программе может потребоваться внесение произвольных записей в */etc/services*.

Подсоединение к сети Ethernet

Теперь, когда IP-адресация понятна, подключиться к сети должным образом будет намного проще. Вероятно, сетевое соединение настроено в ходе начальной установки, однако для внесения изменений в конфигурацию необходимо понимать, как это делается вручную. Для конфигурирования сетевого интерфейса необходима следующая информация:

- Имя сетевого интерфейса
- IP-адрес сервера
- Сетевая маска
- Маршрут по умолчанию

Для конфигурирования сетевой карты применяются две отдельные команды: ifconfig(8) и route(8).[1] ifconfig управляет конфигурацией интерфейса, и если эту команду запустить без аргументов, на экране будут отображены все интерфейсы системы.

```
# ifconfig
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        ether 00:04:5a:41:a4:44
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
lp0: flags=8810<POINTOPOINT,SIMPLEX,MULTICAST> mtu 1500
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
        inet 127.0.0.1 netmask 0xff000000
ppp0: flags=8010<POINTOPOINT,MULTICAST> mtu 1500
```

Интерфейсы представлены вдоль левого края вывода. В системе есть четыре интерфейса: dc0 (Ethernet), lp0 (принтер), lo0 (обратная петля) и ppp0 (точка-точка). Название каждого интерфейса состоит из имени устройства и его номера.

Для получения информации о тот или ином интерфейсе следует обратиться к разделу 4 страниц руководства:

```
# man 4 dc
```

Будем настраивать интерфейс Ethernet. С помощью ifconfig присвоим интерфейсу IP-адрес и зададим сетевую маску:

---

[1] Программа route используется для изменения таблицы маршрутизации, без этой таблицы пакеты не будут передаваться в сеть. Однако к настройке сетевой карты или сетевого интерфейса эта программа никакого отношения не имеет. – *Примеч. науч. ред.*

```
# ifconfig dc0 inet 192.168.1.223 netmask 255.255.255.0
#
```

Программа ifconfig позволяет выяснить конфигурацию отдельного интерфейса.

```
#ifconfig dc0
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.223 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:04:5a:41:a4:44
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
#
```

Заметим, что сетевая маска была преобразована в шестнадцатеричную форму.

Карту Ethernet можно автоматически конфигурировать при начальной загрузке, воспользовавшись параметром файла */etc/rc.conf* (см. главу 8). Запись имеет такой вид: `ifconfig_interfacename="ifconfig statement"`. Например, конфигурация, представленная двумя абзацами выше, в файле */etc/rc.conf* будет выглядеть так:

```
ifconfig_dc0="inet 192.168.1.223  netmask 255.255.255.0"
```

Теперь, когда интерфейс сконфигурирован, попробуем выполнить ping, задав IP-адрес шлюза по умолчанию. Выполнение ping можно прервать, нажав клавиши Control-c. Если ответ получен (как показано в следующем листинге), то машина успешно подключена к сети, а если нет, то дело в плохом соединении или неверной конфигурации карты.

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.631 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.323 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.323/0.477/0.631/0.154 ms
#
```

Маршрут по умолчанию имеет очень простое назначение – по этому адресу система посылает весь трафик, который не может доставить сама. Этот маршрут можно задать с помощью команды route.

```
# route add default 192.168.1.1
```

Вот и все! Теперь ping должен проходить по любому IP-адресу Интернета. Маршрутизатор по умолчанию, устанавливаемый при начальной загрузке, можно задать оператором `defaultrouter` в /etc/rc.conf (см. главу 8). Вот хороший пример оператора `defaultrouter`:

```
defaultrouter="192.168.1.1"
```

Однако при выполнении ping удобнее задавать имена хостов. Если ping по имени не работает, нужно настроить распознаватель. См. в главе 11 раздел,

посвященный файлу */etc/resolv.conf*. Впрочем, такая настройка, вероятно, была проведена в процессе установки системы.

Несколько IP-адресов на одном интерфейсе

Одна система FreeBSD может отвечать на запросы по нескольким IP-адресам на одном интерфейсе. Такая возможность широко применяется в серверах Интернета, особенно для защищенных (secure) веб-сайтов. Возможно, один сервер должен поддерживать сотни или тысячи доменов, и ему нужен IP-адрес для каждого из них. Дополнительные IP-адреса можно добавить с помощью команды ifconfig:

```
# ifconfig dc0 alias 192.168.1.225
```

После запуска предыдущей команды интерфейс будет выглядеть так (исходный IP-адрес всегда идет первым; за ним следуют псевдонимы):

```
# ifconfig dc0
dc0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        inet 192.168.1.223 netmask 0xffffff00 broadcast 192.168.1.255
        inet 192.168.1.225 netmask 0xffffff00 broadcast 192.168.1.255
        ether 00:04:5a:41:a4:44
        media: Ethernet autoselect (100baseTX <full-duplex>)
        status: active
#
```

Дополнительные IP-адреса можно автоматически конфигурировать в ходе начальной загрузки, если задать другую команду ifconfig в файле */etc/rc.conf*:

```
ifconfig_dc0_alias0="inet 192.168.1.225"
```

Единственное реальное различие между этой записью и стандартной записью ifconfig в *rc.conf* – это фрагмент «alias0». Каждый псевдоним, устанавливаемый в */etc/rc.conf*, должен иметь уникальный номер, причем номера должны быть последовательными. Если пропустить какой-нибудь номер, то псевдонимы после пропуска не будут установлены при начальной загрузке. Это самая распространенная ошибка в конфигурировании интерфейсов; система FreeBSD требует перезагрузки так редко, что ошибки в */etc/rc.conf* могут месяцами оставаться незамеченными!

Все исходящие соединения используют действительные IP-адреса. К сетевой карте может быть привязано 2000 IP-адресов, однако когда вы соединяетесь с внешним сервером по ssh, соединение исходит с первичного IP-адреса. Об этом следует помнить при написании правил для брандмауэра и других фильтров, управляющих доступом к системе.

Использование netstat

netstat(1) – это окно в текущую сетевую обстановку. С помощью этой программы можно увидеть состояние соединений, количество сетевых буферов, занятых ядром, и все остальное, что может представлять интерес при работе в сети.

Один из самых важных ключей netstat – ключ -n. По умолчанию netstat показывает имена хостов для каждого соединения, однако выяснение имени хоста отнимает много времени. Ключ -n деактивирует поиск имени хоста по IP-адресу. Если какое-то соединение вызывает особый интерес, имя хоста можно легко определить самому.

Другой важный ключ netstat – ключ -I, который позволяет задавать интерфейс. Некоторые ключи netstat допускают указание конкретного интерфейса или требуют этого. Помните, что в системе есть различные интерфейсы: обратная петля, принтер, Ethernet и т. д.

Ключ -f позволяет выбрать семейство протоколов. Если интерес вызывают только соединения IPv4, задайте ключ -f inet. Другие действительные значения для -f включают в себя inet6 (IPv6), ipx (Novell IPX), atalk (AppleTalk), ng (Netgraph) и unix (сокеты UNIX). В наших примерах можно задавать ключ -f inet, если не указано иное.

Прежде всего рассмотрим существующие соединения:

```
# netstat -na
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp4       0      0  192.168.1.222.22       192.168.1.200.1067     ESTABLISHED
tcp4       0      0  *.5999                 *.*                    LISTEN
tcp4       0      0  *.80                   *.*                    LISTEN
tcp4       0      0  *.443                  *.*                    LISTEN
tcp4       0      0  192.168.1.222.25       *.*                    LISTEN
tcp4       0      0  *.22                   *.*                    LISTEN
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Inode     Conn     Refs  Nextref Addr
d5ba2200 stream      0      0        0 d5ba2240        0        0
d5ba2240 stream      0      0        0 d5ba2200        0        0
...
```

Каждая строка в выводе netstat обозначает сетевое соединение того или иного вида. Можно увидеть довольно много строк, представляющих доменные сокеты UNIX. Это сокеты, которые открыты в ядре, но (пока) не связаны с сетевыми соединениями.[1] Сейчас этим сокетам не стоит уделять внимание; в следующий раз для того, чтобы избежать их появления в выводе netstat, можно задать ключ -f inet.

Первое поле в каждой строке – это протокол. В нашем примере есть только соединения TCP, версия 4 (tcp4).

---

[1] Когда некое приложение, например веб-сервер, запускается и готовится отвечать на запросы по сети, оно выполняет запрос к ядру, требуя «открыть сокет». Тем самым оно сообщает ядру, что намерено «слушать» определенный порт. В случае веб-сервера это обычно порт с номером 80. Когда запрос на порт 80 придет, будет установлено TCP-соединение. Пока запрос не пришел, в ядре имеется открытый, но «незанятый» сокет. – *Примеч. науч. ред.*

Колонки Recv-Q и Send-Q показывают, сколько байт в том или ином соединении ожидают обслуживания. Если в колонке Recv-Q ненулевые значения присутствуют постоянно, значит, система не может обрабатывать входные данные достаточно быстро. Точно так же, постоянно присутствующие значения в Send-Q говорят о том, что либо сеть, либо другая сторона соединения не могут принять данные с той скоростью, с какой они посылаются. Случайные пакеты, ожидающие своей очереди, – это нормально. Для каждой системы надо определить, какая ситуация нормальна, а какая – нет.

Очевидно, что Local Address – это IP-адрес локальной системы. Но приведенные адреса состоят из пяти чисел, разделенных точками! Последнее число – это номер порта. Например, 192.168.1.222.22 – это порт 22 на 192.168.1.222. Если запись представлена как «звездочка, точка, номер порта», это означает, что система прослушивает этот порт на всех IP-адресах. В приведенном примере нет ни одного активного соединения, но система готова «принять» одно соединение.

Колонка Foreign Address (внешний адрес) показывает удаленный адрес (remote address) и номер порта для каждого соединения.

Наконец, колонка (state) показывает состояние рукопожатия TCP. В данный момент не требуется знать все возможные состояния TCP-соединения; достаточно понять, какие из них являются нормальными. Состояние ESTABLISHED означает, что соединение установлено, и данные, по всей видимости, передаются. Состояния LAST_ACK, FIN_WAIT_1 и FIN_WAIT_2 означают, что соединение закрывается. Состояния SYN_RCVD, ACK и SYN+ACK – это этапы нормального создания соединения. В предыдущем примере одно соединение TCP активно. Пять портов TCP ожидают входящих соединений.

Для того чтобы выяснить количество доставленных пакетов, количество пропущенных пакетов и количество ошибок, можно воспользоваться ключом -b. В этом случае вывод netstat будет довольно широким; при работе в среде X лучше раздвинуть текущее окно настолько, насколько позволяет экран. Наиболее интересные колонки: Ierrs (input errors, входные ошибки), Oerrs (output errors, выходные ошибки) и coll (collision, коллизия). Значения в этих колонках должны быть равны нулю или близки к этому. В противном случае необходимо выяснить, в чем причина неполадок. Ошибки могут быть вызваны чем угодно: плохими кабелями, неисправными коммутаторами, сетевыми картами, сбоями в программном обеспечении, ошибками в ПЗУ (firmware errors) и т. п.

С помощью ключа -L можно выяснить, сколько соединений распознала система (будут показаны *очереди ожидания* (*listen queues*)):

```
# netstat -Ln
Current listen queue sizes (qlen/incqlen/maxqlen)
Listen         Local Address
0/0/10         127.0.0.1.1556
0/0/10         127.0.0.1.8080
0/0/10         127.0.0.1.554
0/0/10         127.0.0.1.7070
```

```
0/0/10          192.168.10.6.1556
0/0/10          192.168.10.6.8080
0/0/10          192.168.10.6.554
0/0/10          192.168.10.6.7070
#
```

Каждая строка в этом выводе показывает уникальную пару IP-адрес/порт. Первое число в колонке Listen – это количество непринятых запросов на соединение, которые получила система. Второе число – количество непринятых, незавершенных соединений. Третье – максимальное количество запросов на соединение, которые могут находиться в очереди для того или иного адреса. Завершенное соединение изымается из очереди.

Еще один вариант – команда netstat -m. Он отражает статистические данные об mbufs ядра. Если исчерпаны все mbufs, то система больше не сможет принимать сетевые данные. Они высвобождаются по мере обработки данных.

```
# netstat -m
211/3216/10240 mbufs in use (current/peak/max):
        44 mbufs allocated to data
        167 mbufs allocated to packet headers
41/1114/2560 mbuf clusters in use (current/peak/max)
3032 Kbytes allocated to network (39% of mb_map in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
#
```

Вывод этой команды показывает, что в системе задействовано 39% mbufs. Есть достаточный запас, чтобы справиться с любыми пиками нагрузки. Не было отвергнуто (denied) ни одного запроса на выделение памяти. В случае нехватки mbufs нужно увеличить значение параметра ядра NMBCLUSTERS (см. главу 4).

Ключ -p предоставляет статистические данные для каждого протокола. Вероятно, наибольший интерес вызывают протоколы IP, TCP и UDP. Вывод этой команды очень длинный. Они уникален для каждой системы. Его изучение поможет понять, какие значения являются нормальными для той или иной системы. Любые неполадки оставят свои следы в выводе этой команды.

Для просмотра таблицы маршрутизации надо запустить команду netstat -r:

```
# netstat -r
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use     Netif Expire
default            192.168.1.1        UGSc       10        1      wi0
localhost          localhost          UH          0        2      lo0
192.168.1          link#5             UC          2        0      wi0    794
192.168.1.1        0:a0:cc:35:5b:7    UHLW       12        0      wi0    485
magpire            0:4:5a:41:a4:44    UHLW        1      453      wi0 =>
192.168.87         link#1             UC          0        0     fxp0 =>
#
```

Каждая строка таблицы – это отдельный маршрут. Посылая пакет другому хосту, система FreeBSD обращается к таблице маршрутизации. Заметим, что запуск предыдущей команды отнял некоторое время – netstat пытался найти имя хоста для каждого IP-адреса. Хосты, представленные в виде IP-адреса, не сообщили имена хостов за определенное время. Для получения «быстрой» информации о маршрутизации задайте ключ `-n`.

Первая колонка в предыдущем выводе – `Destination` (назначение). Это может быть хост, сеть или маршрут по умолчанию.

`Gateway` (шлюз) – это место, куда надо отправить пакет для доставки до указанного порта или сети.

Колонка `Flags` (флаги) описывает формирование или использование маршрутов. Полный список всех флагов можно найти в netstat(1), а основные флаги представлены в табл. 5.2. В данный момент необязательно понимать назначение всех этих флагов. Достаточно ознакомиться с флагами для маршрутов, типичных для вашей системы. Если что-то выглядит по-иному, начинайте копать глубже.

Таблица 5.2. Типичные флаги маршрутизации в выводе netstat

Флаг	Описание
`U`	Маршрут пригоден к использованию
`G`	Шлюз
`S`	Статический маршрут (то есть он не добавляется динамически протоколом маршрутизации)
`L`	Этот маршрут – преобразование адреса между протоколами (то есть в строке указан MAC-адрес, используемый для доступа к IP-адресу назначения)
`H`	Маршрут для конкретного хоста
`C`	Маршрут, используемый при динамическом создани новых маршрутов (то есть шлюз)
`c`	Маршрут, который используется для новых маршрутов, специфичных для того или иного протокола (то есть маршруты до шлюза)
`W`	Маршрут, клонированный с другого маршрута

Колонка `Refs` показывает, сколько соединений используют ту или иную запись маршрутизации в выводе `netstat -nr`. В данном примере система использует два маршрута.

Поле `Use` (использование) сообщает, сколько пакетов было послано по этому маршруту.

Колонка `Netif` показывает системный интерфейс, через который доступен маршрут.

Колонка `Expire` (срок действия) сообщает количество секунд, в течение которых маршрут будет считаться истинным. По истечении этого времени систе-

ма проверит, не следует ли обновить маршрут. В данном примере оба маршрута, имеющие ненулевые значения в Expire, находятся в локальной сети Ethernet. Для обновления маршрута система использует стандартный процесс arp.

Наконец, netstat -w показывает текущие статистические данные системы. Данные на экране постоянно обновляются, пока не будет нажата комбинация клавиш Control-c. Для netstat -w надо указать дополнительный аргумент – количество секунд между обновлениями:

```
# netstat -w 5
            input        (Total)           output
   packets  errs      bytes    packets  errs      bytes colls
         1     0         60          1     0        186     0
         1     0         60          1     0        138     0
^C
#
```

Эта информация поможет выяснить, повторяются ли в системе ранее отмеченные ошибки.

6

Обновление FreeBSD

Обновление сервера Интернета доставляет немало хлопот. После обновления программного обеспечения можно смириться с необъяснимым поведением настольного компьютера, однако все должно быть в полном порядке, если от системы зависит деятельность целой компании или сотен потребителей.

Неоднократно я обновлял сервер Windows с NT до 2000 или с 2000 до XP, и каждый раз обнаруживалось, что некоторые функции сервера больше не действуют, как предполагалось. Обновление Linux также может добавить седых волос, а с другими системами UNIX дела обстоят еще хуже. Довольно много администраторов UNIX привычно переустанавливают операционные системы, предпочитая не связываться с обновлением. Несмотря на то что в некоторых версиях UNIX процедуры обновления просты, для их выполнения требуется несколько часов и известная доля удачи. (Последний раз, когда я обновлял HP/UX и базу данных Informix на этой машине, я появился в офисе вечером в пятницу, захватив с собой спальный мешок, будильник и еду, и ушел в полдень понедельника. Я запускал команду и устанавливал будильник на час или два вперед. К этому времени команда выполнялась, и я приступал к следующему этапу.)

Одна из самых сильных черт FreeBSD – процедура обновления. Например, у меня есть несколько серверов, на которых была установлена система FreeBSD 2.2.5 (самая последняя и лучшая на то время.) После этого они были успешно обновлены до 2.2.8, затем прошли путь от 3.0 до последней из версий 3.*, а сейчас на них работает версия 4. Единственное неудобство, которое я испытал, проявилось при переходе от одной основной версии к другой – то есть от FreeBSD 3 к FreeBSD 4. На это я потратил пару часов. Попробуйте сделать то же самое с Solaris, HP/UX или Windows.

Версии FreeBSD

Почему обновлять FreeBSD относительно просто? Все дело в методе разработки FreeBSD. Это постоянно развивающаяся операционная система. Если после обеда вы загружаете определенную версию FreeBSD, она немного от-

личается от утренней версии. Разработчики по всему миру непрерывно вносят изменения и улучшения, поэтому традиционная система нумерации версий, принятая для менее открытого программного обеспечения, здесь будет непрактична. В любой момент можно получить несколько различных версий FreeBSD: release, -current, -stable и моментальную копию.

Release

«*Выпуск*» FreeBSD (*release*) имеет стандартный номер версии, подобно «выпускам» другого программного обеспечения: 2.2.7, 3.3, 4.4, 5.0. Если система FreeBSD куплена в магазине, это «выпуск».

«Выпуск» – это просто копия состояния самой стабильной версии FreeBSD на тот или иной момент времени. Три или четыре раза в год ответственный за «выпуски» (Release Engineer) просит разработчиков приостановить внесение существенных изменений и устранение выявленных неполадок. После тщательного тестирования программный код получает номер «выпуска». Далее разработка снова идет полным ходом, а отдел BSD того или иного поставщика спешит доставить полученный «выпуск» на предприятие, которое воспроизведет его на компакт-дисках.

На важных серверах всегда устанавливайте именно «выпуск».

FreeBSD–current

FreeBSD-current (текущая) – это передний край, самая последняя версия FreeBSD. Она содержит код, который первый раз представляется публике. Версия FreeBSD-current получает первые экспертные оценки и время от времени подвергается радикальным изменениям, которые добавляют забот опытным системным администраторам.

Версия FreeBSD-current доступна для разработчиков, тестеров и заинтересованных сторон, но она не предназначена для всеобщего использования. Ответы на вопросы пользователей о -current очень скудны, поскольку у разработчиков просто нет времени помогать в настройке веб-броузера, когда тысяча более существенных замечаний требуют их внимания. Пользователи должны справиться сами или терпеливо подождать, пока кто-нибудь не устранит эти неполадки.

Если вы не можете читать код C, Perl и командного интерпретатора, или не способны отлаживать ОС, или вам не нравится непредсказуемое выполнение функций, или вам не хочется ждать, пока кто-нибудь устранит возникшие неполадки, то версия -current не для вас.

Храбрецам, желающим испробовать -current, всегда рады. Дорога открыта каждому, кто готов посвятить немало времени изучению и отладке FreeBSD или желает получить урок смирения. Это скорее не наставление «так делать нельзя», а утверждение «все в ваших руках».

Администраторы, выбравшие -current, должны подписаться на почтовые рассылки *FreeBSD-current@FreeBSD.org* и *cvs-all@FreeBSD.org*. Это рассыл-

ки с большим трафиком – несколько сотен предупреждений, извещений и комментариев в день. Читайте их, особенно предупреждения. Если кто-то обнаружил вопиющую ошибку (Bug of Slow Hideous Death), вы должны найти время, чтобы извлечь пользу из его опыта.

Замораживание кода

Каждые год-полтора для FreeBSD-current проводится месячник «замораживания кода» («code freeze»), когда не допускаются несущественные изменения и устраняются все оставшиеся неполадки. По завершении этого процесса (или вскоре после него) -current становится новым «выпуском» .0 системы FreeBSD-stable (стабильной).

Недолгое время, при замораживании кода, версия -current рассматривается как ранний «выпуск» FreeBSD-stable. Внимание разработчиков сосредоточено на устойчивости работы системы и устранении ошибок, выявленных ранее. После одного или двух «выпусков» из новой версии -current ответвляется новая, основная версия -stable. Например, на момент написания этих строк ожидается, что версия 5-current станет версией 5.0-release. Какое-то время после 5.1-release у версии -current будет оставаться номер 5.0, чтобы обратить внимание разработчиков на новый «выпуск». Спустя определенное время после 5.1-release копия исходного кода будет обозначена как 6.0-current, а другая копия – как 5.1-stable.

FreeBSD–stable

FreeBSD-stable – это передний край для среднего пользователя. Эта версия содержит самый последний код, аттестованный экспертами. Версии FreeBSD-stable положено быть устойчивой и надежной; она не должна требовать значительного внимания пользователя. Как только тот или иной код в -current тщательно протестирован, он может влиться в версию -stable (рис. 6.1). Этот процесс называется *поглощением из текущей версии* (*merge from current*, *MFC*). На версию -stable можно безопасно перейти почти в любое время; это своего рода FreeBSD-beta.

Со временем различия между -stable и -current становятся все больше и больше и в какой-то момент назревает необходимость ответвить от -current новую версию -stable. Предыдущая -stable будет активно поддерживаться несколько месяцев, пока не закрепится новая -stable.

Одни пользователи захотят немедленно обновить систему до новой версии -stable, а другие будут более осмотрительны. После выхода одной или двух новых версий -stable предшествующая версия -stable признается устаревшей, а пользователи увидят приглашение обновить системы до новой -stable. В конце концов, изменения в устаревшей версии -stable будут сводиться лишь к исправлению критических ошибок.

Время от времени на версию -stable «наводят лоск» и подвергают ее испытаниям; разработчики останавливают перенос изменений из -current в -stable

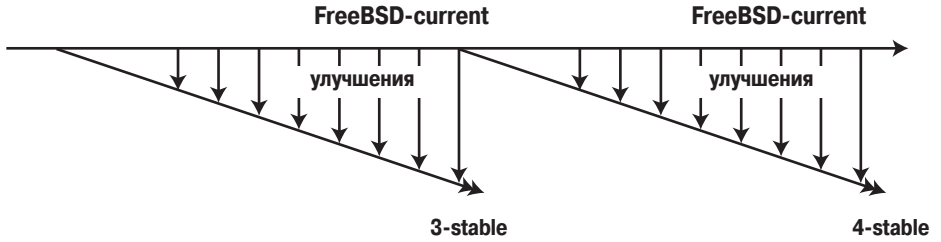

Рис. 6.1. *Направления разработки FreeBSD*

(MFC)[1] и все внимание обращают на тестирование. Когда каждый разработчик удовлетворится качеством версии, подготавливается новый «выпуск». Обычно он получает номер, образованный добавлением «точки» и уточняющего числа к «основному» номеру. (Например, четвертый «выпуск» FreeBSD 4 – это FreeBSD 4.4. В списке версий можно увидеть ссылки на 4-stable и на 4.4-stable – в группу 4-stable включены все «выпуски» 4.x и ответвления -stable.)

Слово *stable* (*стабильный*) описывает основной код, а не ОС в целом. Нет никакой гарантии, что все компоненты операционной системы стабильны и надежны, однако базовый код не будет подвергаться радикальным изменениям. Например, многие люди считают, что FreeBSD 3.5-stable более надежна, чем FreeBSD 4.0-stable.

Примечание

Возможно, FreeBSD является одной из самых надежных операционных систем, однако следует остерегаться любых «выпусков» .0, о какой бы системе ни шла речь. Помните бедных малых, установивших ОС Windows 2000 в течение месяца со дня ее выхода?

Пользователям FreeBSD-stable следует подписаться на почтовую рассылку FreeBSD-stable, трафик которой модерируется. Важные сообщения от разработчиков помечаются словами «HEADS UP» в заголовках писем. Изучайте эти сообщения и следуйте всем приведенным рекомендациям.

Моментальные копии

Время от времени команда разработчиков FreeBSD выпускает моментальные копии (snapshots) версии -current (они доступны через FTP и на компакт-дисках). Моментальная копия не предполагает такого внимания к качеству, как -release. Она служит хорошей отправной точкой для людей, проявляющих интерес к исследованию и тестированию -current. Вообще говоря, разработчики стараются не добавлять существенные функции в последнюю неделю перед созданием моментальной копии, однако ее качество не подвергается анализу. В ней есть ошибки; большинство из них извест-

[1] MFC (сокр. от merge from current) используется в протоколах изменений CVS, фиксируя перенос изменения из ветви current в ветвь stable. – *Примеч. науч. ред.*

но, но далеко не все. Новые функции реализованы не полностью. Эту версию можно назвать самой последней.

Обновления, связанные с безопасностью

С появлением FreeBSD 4.3 Проект FreeBSD стал поддерживать специальные варианты, отличающиеся повышенным вниманием к безопасности. До этого пользователь FreeBSD должен был обновлять систему до последней версии -stable, чтобы установить «заплатки безопасности» (security patches). Однако это вызывало сложности, так как ОС изменялась между «выпусками». Для чего обновлять весь сервер и преодолевать сопутствующие трудности, если надо установить всего лишь заплатку, решающую маленькую проблему безопасности? (Каждый пользователь, устанавливающий обновление Windows 2000 Service Pack, может засвидетельствовать, насколько труден этот процесс.) Устранены только прорехи в безопасности и серьезные системные ошибки; новые функции не добавлены, а производительность не улучшена. Это очень скромный вариант версии -stable.

Названия обновлений, относящихся к обеспечению безопасности, образованы названием выпуска и замыкающим номером заплатки. Например, 4.3-RELEASE-p6 – это шестая заплатка для 4.3-RELEASE.

Какой «выпуск» использовать?

Для разграничения вариантов использования FreeBSD разработчики применяют ту же систему «выпусков», что и для управления качеством. Такая система может показаться сложной, но благодаря ей пользователь может быть уверенным, что тот или иной «выпуск» поддерживается определенным сообществом, прошел экспертную оценку и всестороннее тестирование. Тот же самый пользователь знает, что отдельные новые функции в -stable и -current можно приобрести за определенную сумму.

Итак, какой выбрать «выпуск»?

- **Production** – если ваша система обеспечивает работу предприятия, остановитесь на «безопасном» варианте -release.
- **Test** – сетевым администраторам, желающим узнать, как изменения в FreeBSD влияют на операционную среду, следует применять -stable на испытательной системе.
- **Development** – если вы – невозмутимый разработчик операционной системы, или обладаете массой времени, или круглый дурак, то вариант -current – для вас. Когда -current уничтожит вашу коллекцию MP3, разберитесь, в чем тут дело, и предложите свою заплатку.
- **Hobby** – если вы – увлекающийся человек, запускайте любую версию! Просто помните об ограничениях того варианта, который вы выбрали. Новичкам в UNIX лучше выбрать -release. Как только почувствуете себя увереннее, обновите систему до -stable. Если у вас нет занятия получше и вы испытываете глубочайшее презрение к своим данным, добро пожаловать в ряды мазохистов, выбравших -current.

Методы обновления

В ситуации со всеми этими «выпусками» обновление всегда важно. Есть два основных способа обновить систему: sysinstall и CVSup.

Новички и неопытные пользователи могут применять программу sysinstall, которая позволяет обновить систему только до -release. Опытные пользователи, возможно, выберут CVSup и *make world*. Такой вариант позволяет обновить систему до текущей, самой последней версии FreeBSD, будь то -stable, «безопасная» версия или -current, однако CVSup требует больших усилий с точки зрения установки и применения.

При обновлении с одной версии -release до другой или до моментальной копии можно использовать sysinstall. Пользователи, выбравшие -current, -stable или «безопасные» варианты, должны уметь собирать систему на основе исходного кода (как описано в разделе «Обновление через CVSup» этой главы).

Примечание

Перед обновлением имеет смысл потратить время на полное резервирование системы. Хотя sysinstall ошибается в «лучшую» сторону, при неправильном проведении обновления файлы могут быть потеряны. Для получения инструкций по резервному копированию обратитесь к главе 3. (Конечно, если вы читаете эту книгу последовательно, глава 3 уже изучена.)

Обновление через sysinstall

Самый простой способ обновить систему через sysinstall состоит в том, чтобы загрузиться с установочной дискеты или с компакт-диска с версией системы, до которой происходит обновление. Далее надо пройти следующие этапы:

1. Когда на экране появится графическое меню установки, выберите пункт Upgrade (Обновление).
2. Sysinstall откроет заметки по обновлению, относящиеся к установленной версии sysinstall. (Прочтите их внимательно, потому что в них будут описаны все проблемы, выявленные в последний момент. Кроме того, изучите список ошибок, доступный по адресу *http://www.FreeBSD.org*. Внимательно следуйте инструкциям.)
3. На экране появится вопрос – продолжать ли процесс обновления. На положительный ответ последует вопрос о дистрибутивах. В этот момент полезно знать, какие дистрибутивы были установлены в системе, поскольку, возможно, их придется заменить. Как минимум, необходимо заменить дистрибутив *bin*. Можно здорово запутаться, не обновив все, что было установлено первоначально; недоумение человека, работающего с программами из 4.7-release по документации из 4.3-release, может быть бесконечным.
4. Обновления во многом похоже на установку – до того момента, когда будет задан вопрос о каталоге, в котором надо сохранить резервную копию текущего каталога */etc*. (Помните, что в */etc* содержится большая часть данных о конфигурации системы, и оригиналы конфигурационных фай-

лов необходимо хранить на случай неполадок.) Обычно для этого вполне подходит каталог */usr/tmp/etc*, предлагаемый по умолчанию.

5. Наконец, sysinstall спросит, откуда следует взять исходные файлы для обновления. Можно использовать FTP, компакт-диск с FreeBSD и любые другие методы.
6. Далее sysinstall предоставит последнюю возможность отказаться от обновления, а затем перезапишет все указанные исполняемые файлы системы. Ядро будет заменено ядром GENERIC новой версии; кроме того, будут заменены многие файлы в */etc*.
7. После обновления внимательно просмотрите */etc* и убедитесь, что важнейшие системные файлы соответствуют предъявляемым к ним требованиям. Хотя файлы с паролями и списками групп, а также таблица файловых систем останутся невредимыми, стоит проверить *rc.conf*, *inetd.conf*, оболочки (shells) и любые другие измененные файлы. (Это наиболее утомительная часть процесса обновления.) Если в системе есть файлы с исходным кодом, этот процесс можно облегчить с помощью mergemaster(8). (mergemaster рассматривается в разделе «Изменения */etc* и */dev*» далее в этой главе.)

   После следующей перезагрузки система будет благополучно обновлена.

Примечание

Не используйте sysinstall, который включен в текущую версию FreeBSD! Если вы работаете с системой FreeBSD 4.4-release и хотите обновить ее до 4.5-release, используйте программу sysinstall из состава 4.5-release. Самый простой путь – загружаться с установочной дискеты или с установочного компакт-диска с версией 4.5-release.

Обновление через CVSup

Если вам нужен более гибкий способ обновления системы, попробуйте задействовать исходный код и CVSup. Модификации FreeBSD, выпускаемые различными разработчиками, становятся доступными на серверах FreeBSD по всему миру. Благодаря CVS и CVSup это занимает не более 66 минут. Такой скоростью внесения изменений могут похвастаться только операционные системы BSD. Основной (master) CVS-сервер FreeBSD отслеживает исходный код. На этом сервере воспроизводятся все изменения вместе с указанием их авторов; разработчики могут «зарегистрировать» (check in) новый код, а пользователи – «захватить» (check out) последние версии.

Система параллельных версий (Concurrent Versions System, CVS) – подходящий инструмент для управления исходным кодом, совершенно не пригодный для его распространения; CVS предъявляет большие требования к системным ресурсам и полосе пропускания и, похоже, стремится вывести из строя жесткий диск сервера.

Поскольку все ресурсы Проекта FreeBSD поступают от кого-либо в дар, их необходимо использовать как можно эффективнее. Так, в Проекте FreeBSD для распространения исходного кода вместо CVS применяется CVSup. CVSup – это комбинация CVS и *sup* (Software Update Protocol – протокол обновления

программного обеспечения). По сравнению с CVS, протокол CVSup намного быстрее, эффективнее, «легче» для серверов и, как правило, удобнее для поддержки миллионов пользователей, разбросанных по всему миру. Информация с основного CVS-репозитария исходного кода дублируется на серверах CVSup по всему миру, а пользователи с помощью CVSup подключаются к серверам и загружают исходный код.

Поскольку эти изменения открыто поддерживаются посредством комбинации «CVS/сервер CVSup», любая машина FreeBSD может подключиться к серверу CVSup, сравнить свою локальную копию исходного кода FreeBSD с версией, доступной на сервере, и скопировать все изменения на локальный жесткий диск. На словах это кажется сложным, но на самом деле процесс очень прост. Достаточно установить CVSup на локальной системе и использовать его для эффективной загрузки обновлений.

Установка CVSup

В отличие от большей части FreeBSD, написанной на C, CVSup написан на языке Modula-3. Modula-3 – это очень мощный, современный язык программирования, который хорошо подходит для создания таких приложений, как CVSup. Чтобы собрать CVSup из исходного кода, необходимо сначала собрать Modula-3, что займет какое-то время. Более того, Modula-3 скорее всего больше не понадобится, потому что он необходим лишь для очень немногих программ. Однако если вы последовали рекомендациям, приведенным в главе 1, то CVSup в системе уже установлен.

Если же это не так, его можно установить из пакета, содержащего предварительно скомпилированную версию, или по FTP. В последнем случае необходимы наличие работающего соединения с Интернетом и возможность связи с остальным миром по FTP. Далее нужно ввести следующие команды (как пользователь root):

```
# cd /usr/ports/net/cvsup
# make all install clean
```

По экрану пробегут многочисленные сообщения компилятора, последнее из них подтвердит установку CVSup. Закончив установку, убедитесь в наличии в системе исходного кода FreeBSD – вы должны увидеть примерно следующее:

```
# ls /usr/src
COPYRIGHT         contrib          release
CVS               crypto           sbin
Makefile          etc              secure
Makefile.inc0     games            share
Makefile.inc1     gnu              sys
Makefile.upgrade  include          tools
README            kerberosIV       usr.bin
UPDATING          lib              usr.sbin
bin               libexec
#
```

В выводе команды ls отображено дерево исходного кода FreeBSD; это весь исходный код, необходимый для сборки программ и ядра FreeBSD. (Исходный код подробно рассматривается в главе 10.) При желании можно просмотреть эти каталоги и выяснить, что собой представляет исходный код.

Если этот каталог пуст, значит, исходный код не установлен. Не беда. Исходный код можно инсталлировать с установочного компакт-диска. Для этого надо выполнить следующие команды (войдя в систему с правами root):

```
# mount /dev/acd0c /cdrom
# cd /cdrom/src/
# ./install.sh all
```

Если установочного компакт-диска нет, исходный код можно взять с FTP-сервера FreeBSD или, при хорошей пропускной способности, просто запустить CVSup даже без исходного кода на диске. CVSup сравнит то, что есть, с тем, что должно быть, и установит последнюю версию исходного кода. (Однако администраторы, обслуживающие зеркала CVSup, предпочитают, чтобы желающие установить исходный код брали его с компакт-диска; для поддержки такого сервиса они предоставляют часть своей полосы пропускания и времени процессора, а это недешево.) Весь исходный код занимает около 300 Мбайт.

Какой бы метод установки исходного кода вы ни избрали, отправной точкой будет исходный код для уже установленной версии FreeBSD. Например, компакт-диски FreeBSD 4.5 содержат исходный код для FreeBSD 4.5. Если исходный код инсталлируется при установке системы, будет установлен исходный код для версии 4.5. Исходный код – это полезный источник для программиста. В то же время для выполнения обновления требуется совсем не это; если для перекомпоновки и переустановки FreeBSD будет применен исходный код FreeBSD 4.5, то эта версия будет установлена заново! CVSup сравнивает исходный код на диске и исходный код, доступный в Интернете, а затем загружает код, отражающий изменения между версиями. Далее CVSup «приложит» эти отличия (diffs) к исходному коду на диске и приведет его в соответствие с исходным кодом необходимой версии. Это намного более эффективно, чем загружать все 300 Мбайт! Если между проведением обновлений пропущен один или два «выпуска», CVSup загрузит только один или два мегабайта, необходимых для изменения исходного кода на диске.

Применяя CVSup для обновления исходного кода, надо указать, что обновлять, с какого источника и как.

Выбор supfile

CVSup использует *config*-файл, или *supfile*, который задает файлы, подлежащие обновлению, и версию FreeBSD, до которой их надо обновить. (Простые *supfiles* можно найти в */usr/share/examples/cvsup/*.) Supfile изменяется в соответствии с тем исходным кодом, который необходимо обновить. Файл, созданный для отслеживания изменений в -stable, -current или в «безопасных» вариантах, можно использовать многократно.

Последняя версия каталога */usr/share/examples/cvsup* содержит следующие supfiles:

cvs-supfile

Позволяет загружать весь репозитарий исходного кода FreeBSD. Большинство пользователей в этом не нуждаются, но разработчики FreeBSD находят такую возможность удобной. Этот файл необходим лишь тем, кто подготавливает собственные «выпуски» системы.

doc-supfile

Позволяет извлечь всю документацию FreeBSD на всех доступных языках, включая последнюю версию FAQ и Справочник. Этот файл пригодится лишь при намерении установить */usr/ports/textproc/docproj* и собрать документацию из исходных текстов. Компоновка документации происходит быстрее, чем сборка самой системы FreeBSD, однако сборка docproj займет какое-то время.

gnats-supfile

Предназначен для тех, кто хочет иметь локальную копию базы данных с сообщениями о неполадках во FreeBSD (problem report (PR) database). Большинству пользователей она не нужна.

ports-supfile

Его можно использовать для обновления дерева «портов»[1] (ports) до последней версии.

stable-supfile

Позволит обновить исходный код до последней версии -stable.

standard-supfile

Позволит обновить исходный код до последней версии -current.

www-supfile

Предназначен для загрузки последней версии веб-сайта FreeBSD.

Различные компоненты, которые можно обновить с помощью CVSup, называются *коллекциями* (*collections*). Например, есть коллекция исходного кода, коллекция документации (doc-supfile), коллекция «портов» (port-supfile) и т. д. Многие коллекции разбиты на подколлекции: дерево исходного кода включает в себя подколлекции для таких компонентов, как программы userland, компиляторы, ядро и т. д. При обновлении FreeBSD главный интерес представляет коллекция исходного кода.

---

[1] Ports, или «порты», – так называемые «портированные приложения». По сути представляют собой заранее адаптированные к FreeBSD программы, написанные для использования в любых системах UNIX. Установка уже готовых, адаптированных программ экономит время. Однако ports неудобно использовать, если пытаешься установить новую программу в старую систему, ибо код свежих ports рассчитан на свежую ОС. – *Примеч. науч. ред.*

Модификация supfile

Выбранный supfile необходимо модифицировать в соответствии с предъявляемыми требованиями. Прежде всего скопируйте образец supfile в */usr/src* и откройте его в удобном редакторе.

Строки, начинающиеся со знака «решетки» (#), представляют собой комментарии. Во всех образцах supfiles комментариев больше, чем действительных конфигурационных записей. Большинство supfiles содержат по крайней мере шесть записей, подобных записям в нынешнем stable-supfile:

```
❶ *default host=CHANGE_THIS.FreeBSD.org
❷ *default release=cvs ❸ tag=RELENG_4
❹ *default base=/usr
❺ *default prefix=/usr
❻ *default delete use-rel-suffix
*default ❼ compress
❽ src-all
```

Прежде всего необходимо выбрать сервер CVSup (❶). (Полный список серверов можно найти по адресу *http://www.FreeBSD.org/handbook/mirrors-cvsup.html*.) На момент написания этих строк список серверов начинается с *cvsup1.FreeBSD.org* и заканчивается *cvsup17.FreeBSD.org*, однако он постоянно пополняется новыми серверами, поэтому его надо изучить. Выполните ping до каждого сервера, чтобы найти «ближайший», – самый быстрый отклик ясно свидетельствует, что данный сервер расположен «ближе». Его имя нужно вписать в строку `default host` (хост по умолчанию), заменив `CHANGE_THIS.FreeBSD.org`.

Строка `default release` («выпуск» по умолчанию) – это метка выбранной версии или коллекции (❷). Метку для выбранного варианта поместите в оператор `tag` (❸). `RELENG_4` – это последняя версия 4-stable. Вот некоторые теги (tags), которые можно использовать:

Тег	«Выпуск»
`RELENG_4`	FreeBSD 4-stable
`RELENG_3`	FreeBSD 3-stable
`.`	FreeBSD-current
`RELENG_4_3`	Обновления, относящиеся к безопасности (только для FreeBSD 4.3)
`RELENG_4_4`	Обновления, относящиеся к безопасности (только для FreeBSD 4.4)
`RELENG_4_5`	Обновления, относящиеся к безопасности (только для FreeBSD 4.5)

Строка `default base` (база по умолчанию) – это место, в котором CVSup хранит файлы состояния, включая список обновленных файлов. Эта информация ускорит будущие обновления (❹).

Строка `default prefix` (префикс по умолчанию) – это место, куда будет помещена выбранная коллекция. Выбрав значение по умолчанию, вы не ошибетесь (❺). Для установки исходного кода в каталог, отличный от */usr/src*, необходимо изменить этот путь.

Строка `default delete` (удалять по умолчанию) дает CVS права на удаление устаревших, ненужных исходных файлов (❻). Запись `use-rel-suffix` позволяет CVSup задействовать общий базовый каталог для нескольких версий исходного кода, не смешивая их.

Все примеры supfiles включают в себя инструкцию `defaut compress` – сжимать данные, передаваемые с помощью CVSup (❼). Если соединение организовано по T1 или более быстрому каналу, сжатие данных не настолько важно и эту строку можно удалить. Нагрузка на процессор при этом снизится, но потребуется более широкая полоса пропускания. Современные процессоры обычно дешевле аренды полосы пропускания, поэтому от сжатия отказываются довольно редко.

Тег src-all предписывает CVSup обновлять все дерево исходного кода (❽). В файле *stable-supfile* есть список закомментированных подколлекций, таких как *usr.bin*, *contrib*, *sys* и т. д. Все они включены в *src-all*. Чтобы обновлять только часть исходного кода, надо задать соответствующие названия подколлекций. Например, для обновления исходного кода в */usr/src/usr.bin* следует указать подколлекцию usr.bin. Использовать для обновления системы лишь этот исходный код – совершенно никудышная идея. Например, установка каталога */usr/bin* системы 4.5-release на системе 4.3-release повлечет за собой непредсказуемые сбои, а значит, не поддерживается.

Указание нескольких коллекций

В одном supfile можно указать несколько коллекций. Например, мне нужен доступ к коллекции исходного кода для последней версии -stable. Как ответственному за документацию FreeBSD, мне нужна последняя коллекция документации. Плюс ко всему мне необходимо последнее дерево «портов», чтобы можно было устанавливать самое последнее программное обеспечение.

В состав FreeBSD входят отдельные примеры supfiles для каждой из этих коллекций. Так как мне нужно все сразу (я не хочу запускать CVSup три раза для загрузки всех последних изменений), я применяю один supfile для получения последней подходящей версии FreeBSD, последней коллекции «портов» и последней коллекции документации:

```
*default host=cvsup16.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default delete use-rel-suffix
*default compress src-all
*default release=cvs tag=RELENG_4
src-all
❶ ports-all tag=.
doc-all tag=.
```

Все записи до предпоследней строки (❶) – это записи стандартного stable-supfile. После них можно указать любые дополнительные коллекции.

Особый интерес вызывают последние две записи, в которых добавлены коллекции `ports-all` и `doc-all`. Коллекции, не относящиеся к исходному коду,

не имеют «выпусков» и вариантов, которые есть у коллекций исходного кода. Значит, если указать версию RELENG_4 дерева «портов», сервер не поймет, что от него хотят, и обновления не будут загружены. Если к названию коллекции дописать ключевое слово "tag=.", CVSup загрузит последнюю версию этой коллекции.

Дерево «портов» будет обсуждаться в главе 10.

Блокирование обновлений CVSup с помощью файла refuse

Для того чтобы отказаться от обновления определенных программ, создайте файл *refuse*. Например, я постоянно обновляю свою коллекцию портов, а значит, могу легко получить самое последнее программное обеспечение. Однако некоторые категории в коллекции портов не представляют для меня интереса, например неанглоязычные программы. В каталоге */usr/ports* есть подкаталоги для программного обеспечения на китайском, немецком, японском, корейском, русском и вьетнамском языках. Вероятность того, что эти пакеты мне потребуются, близка к нулю. Чтобы предписать CVSup не обновлять эти каталоги, создайте файл */usr/sup/refuse*, который выглядит примерно так:

```
ports/chinese
ports/german
ports/japanese
ports/korean
ports/russian
ports/vietnamese
```

Примечание

В файле *refuse* не должно быть комментариев!

Файл *refuse* можно создать для любых компонентов FreeBSD, однако лучше не указывать в нем что-либо из */usr/src*. Если отказаться от обновления существенной системной программы, эта программа в какой-то момент станет несовместима с системой.

Система интерпретирует файл *refuse* на основе сопоставления подстрок. Строка "sys" блокирует обновление всех программ, содержащих «sys», и в том числе исходный код ядра в */usr/src/sys*. Будьте внимательны с файлом *refuse* – его строки должны блокировать только то, что необходимо блокировать!

Примечание

Если в supfile префикс */usr* изменен и указывает на другое местоположение, то необходимо переместить и файл *refuse*. (Его действительное местоположение – *$PREFIX/sup/refuse*.) Если принять значения по умолчанию, приведенные примеры будут работать.

Обновление исходного кода системы

После создания supfile зайдите в систему под учетной записью root и запустите CVSup, введя следующую команду:

```
# cvsup supfile
```

В среде X программа CVSup откроет GUI, в командной строке она начнет обновлять исходные файлы. После выполнения команды в системе появится самая последняя версия исходного кода FreeBSD.

Частичные обновления

CVSup поддерживает частичные обновления с помощью параметра `-i 'pattern'`. Например, `cvsup my-supfile -i src/etc` обновит каталог */usr/src/etc*, используя настройки по умолчанию из *my-supfile*. Для получения кода ядра можно выполнить `cvsup my-supfile -i sys`, а для загрузки последнего демона FTP подойдет `cvsup my-supfile -i ftpd`.

Однако при выполнении обновлений будьте внимательны. Система FreeBSD задумана и работает как одно целое, поэтому нельзя смешивать и совмещать различные версии утилит. Например, если новая версия демона FTP опирается на функции ядра, добавленные в последний месяц, этот демон не будет работать на предшествующем ядре.

Примечание

CVSup заботится лишь об обновлении базовой части операционной системы FreeBSD. Отдельные программы, такие как оболочки, веб-серверы или редакторы, надо обновлять вручную. Подробности приведены в главе 9.

После обновления исходного кода

После обновления исходного кода изучите файл */usr/src/UPDATING*. В этом файле в обратном хронологическом порядке приведены все предупреждения и замечания, относящиеся к изменениям в исходном коде. Здесь указаны действия, которые, возможно, придется выполнить перед сборкой системы из исходного кода, а также описаны любые существенные изменения в функциональности системы.

Кроме того, стоит ознакомиться с новыми конфигурационными файлами *GENERIC* или *LINT*. В них могут быть добавлены новые параметры, а в ядре могут появиться небезынтересные изменения.

Сборка новой системы FreeBSD

После того как программа CVSup обновит исходный код, можно пересобрать систему, причем по-разному. Можно услышать веские доказательства преимуществ того или иного метода по сравнению с другими в скорости, надежности и качестве его работы. Метод, рекомендуемый командой Проекта FreeBSD, – самый безопасный. Крайне маловероятно, что он причинит вред системе. Кроме того, это замечательный кандидат для script(1) (script рассматривается в главе 3).

Единственный рекомендованный способ компоновки FreeBSD подразумевает выполнение следующих команд:

```
# cd /usr/src
# make buildworld
```

Команда `make buildworld` прежде всего задействует исходный код, чтобы собрать инструменты, необходимые для сборки системного компилятора. После этого она инициирует сборку компилятора и сопутствующих библиотек. Наконец, с помощью новых инструментов, компилятора и библиотек будет собрано все базовое программное обеспечение FreeBSD. (Однако эти компоненты не установятся, а будут помещены в */usr/obj* для дальнейшего использования.)

Выполняя `make buildworld`, система FreeBSD пересобирает буквально каждый компонент самой себя, что в зависимости от аппаратного обеспечения может занять от одного до нескольких часов. При этом можно продолжать работать (хотя команда `make buildworld` потребляет системные ресурсы, она не требует к себе внимания). Однако на устаревшем компьютере она выполнится куда медленнее.

Этот процесс можно оптимизировать, внеся изменения в */etc/make.conf* (см. главу 9). Стоит обдумать применение записи `CPUTYPE` и различных флагов `NO_`. Таким образом можно создать более эффективную операционную систему, а обновление пройдет более быстро и гладко.

Примечание

Убедитесь, что команда `make buildworld` выполнилась без ошибок! Если она завершилась выдачей сообщений об ошибках (error-code), похожих на сообщения при неудачном компилировании ядра, то процесс надо немедленно остановить. В главе 2 рассказано, как получить помощь. Ни при каких обстоятельствах не пытайтесь установить поврежденные или неполные обновления.

Обновление ядра

Если команда `make buildworld` выполнена успешно, можно приступить к обновлению ядра. Здесь есть ловушка: последние исполняемые файлы инструментов, необходимых для сборки ядра, на самом деле не установлены и располагаются в другом дереве каталогов! Стандартная процедура `config`, `make depend`, `make install`, рассмотренная в главе 4, в этом случае не подойдет.

Если ранее была разработана собственная конфигурация ядра, необходимо в */etc/make.conf* установить переменную `KERNCONF` (см. главу 8). В противном случае будет заново собрано ядро GENERIC.

Для сборки ядра надо, оставаясь в */usr/src*, запустить следующую команду:

```
# make buildkernel
```

Примечание

Для обеспечения максимальной безопасности запускайте `make buildkernel KERNCONF=GENERIC`. В этом случае в системе появится обновленное ядро *GENERIC*.

Установка системы

После выполнения `make buildkernel` можно приступить к установке заново собранной системы. Первый шаг – установить новое ядро и модули ядра.

Из всех компонентов, подвергшихся обновлению, легче всего протестировать новое ядро и сопутствующие модули. В случае неполадок можно просто вернуться к старому ядру. Установите новое ядро и модули следующей командой:

```
# make installkernel
```

Для тестирования нового ядра перезагрузите систему и войдите в нее. Все ли программы работают? Получает ли система почту? Обслуживает ли веб-страницы? Если так, значит, с ядром, по всей видимости, все в порядке. Различные программы, которым необходим доступ к структурам ядра, такие как top(1), не будут работать, но можно протестировать все остальное.

Если ядро и программы работают правильно, можно устанавливать оставшуюся часть системы. Установка несистемных приложений (userland) должна быть выполнена в однопользовательском режиме. Если система работает на уровне безопасности 1 (securelevel 1) и выше, надо остановить систему и загрузиться в однопользовательском режиме. Кроме того, для перехода в однопользовательский режим можно просто выполнить такую команду:

```
# shutdown now
```

Как только система перешла в однопользовательский режим, можно устанавливать программы userland:

```
# cd /usr/src
# make installworld
```

При выполнении этих команд будут установлены различные программы, составляющие базовую часть системы FreeBSD. По экрану побегут многочисленные сообщения, в основном со словом «install». Окончание этого процесса означает установку системы. Однако для полного завершения обновления необходимо выполнить еще несколько шагов.

Изменения /etc и /dev

Как известно, файлы конфигурации, специфичные для вашей системы, находятся в */etc*. Поскольку никакая программа не знает о том, как вы считаете нужным запускать вашу систему, необходимо вручную внести все изменения в файлы */etc*. Команды с `make buildworld` до `make installworld` на это не способны.

Последние версии всех файлов и каталогов */etc* находятся в */usr/src/etc*. Эти файлы (особенно файлы «*rc*») необходимо сравнить с файлами в */etc* и выяснить, какие изменения следует внести. Делать это вручную утомительно и сложно, поэтому воспользуемся утилитой mergemaster(8), что позволит значительно сэкономить время, требуемое для объединения файлов.

Чтобы воспользоваться mergemaster, войдите в систему с правами root и наберите следующую команду:

```
# mergemaster
```

mergemaster скопирует различные файлы настройки из дерева исходного кода, создаст временные каталоги */etc* и */dev* в */var/tmp*, а затем сравнит эти файлы с файлами настройки в существующем */etc*. Если новая версия файла отличается от версии в */etc*, mergemaster отобразит различия. В этот момент можно решить: оставить старый файл, использовать новый или объединить их.

Вот, в качестве примера, начало типичной сессии mergemaster:

```
*** Beginning comparison (начинаем сравнение)

❶ *** Temp ./dev/MAKEDEV.local and installed have the same $FreeBSD, deleting
      (временный файл ./dev/MAKEDEV и установленный файл имеют одинаковое значение
       $FreeBSD, удаление)
❷ *** Temp ./dev/MAKEDEV and installed have the same $FreeBSD, deleting

  *** Displaying differences between ./etc/defaults/rc.conf and installed version
      (Отображение различий между ./etc/defaults/rc.conf и установленной версией)

❸ --- /etc/defaults/rc.conf       Fri Jan 14 08:54:11 2000
+++ ./etc/defaults/rc.conf      Sun Feb 27 18:21:29 2000
@@ -9,7 +9,7 @@
 # # All arguments must be in double or single quotes.
     (Все аргументы должны быть в двойных или одинарных кавычках).
 #
-# $FreeBSD: src/etc/defaults/rc.conf,v 1.1.2.18 1999/11/28 16:02:30 brian Exp $
+# $FreeBSD: src/etc/defaults/rc.conf,v 1.1.2.19 2000/02/19 13:11:28 jkh Exp $

 ##############################################################################
 ### Important initial Boot-time options (Важные опции начальной загрузки) ########
@@ -193,7 +193,7 @@
 saver="NO"               # screen saver: Uses /modules/${ saver} _saver.ko
                            (хранитель экрана)
 moused_enable="NO"       # Run the mouse daemon. (запуск демона мыши)
 moused_type="auto"       # See man page for rc.conf(5) for available settings.
                   (см. доступные настройки на страницах руководства по rc.conf(5))
-moused_port="/dev/cuaa0" # Set to your mouse port. (порт мыши)
+moused_port="/dev/psm0"  # Set to your mouse port.
 moused_flags=""          # Any additional flags to moused.
                            (дополнительные флаги для мыши)
 allscreens_flags=""      # Set this vidcontrol mode for all virtual screens
                            (установить режим vidcontrol для виртуальных экранов)

  Use 'd' to delete the temporary ./etc/defaults/rc.conf
  (для удаления временного ./etc/defaults/rc.conf используйте 'd')
  Use 'i' to install the temporary ./etc/defaults/rc.conf
  (для установки временного ./etc/defaults/rc.conf используйте  'i')
  Use 'm' to merge the old and new versions
  (для объединения старой и новой версий используйте 'm')

  Default is to leave the temporary file to deal with by hand
  (по умолчанию – редактировать временный файл вручную)

How should I deal with this? [Leave it for later]
(Как мне поступить? [Оставить на потом])
```

Посмотрим, что здесь есть. Первые две строки (❶ и ❷) показывают, что версия */dev/MAKEDEV* в */dev* и версия файла в последнем дереве исходного кода совпадают. mergemaster не предлагает вносить изменения, а просто удаляет временные копии, созданные для сравнения файлов.

Далее mergemaster переходит к измененному файлу – */etc/defaults/rc.conf* (❸). mergemaster выдает несколько строк файла; добавления помечены знаками «плюс» (+) в начале строки, а удаленные строки – дефисами (-). (Несколько строк выше и ниже приводятся для контекста.)[1]

Рассмотрим такие строки:

```
-moused_port="/dev/cuaa0" # Set to your mouse port. (порт мыши)
+moused_port="/dev/psm0"  # Set to your mouse port.
```

В них сообщается, что порт мыши, заданный по умолчанию, изменен в период между текущим и предшествующим обновлением (или установкой). Строки, помеченные знаком «минус», находятся в старой версии файла, а строки, помеченные знаком «плюс», – в новой версии. Порт был */dev/cuaa0* (последовательный порт), а теперь стал */dev/psm0* (порт PS/2).

mergemaster предложит три варианта: удалить, установить или объединить. В данном случае надо установить новый файл */etc/defaults/rc.conf* и внести необходимые изменения в */etc/rc.conf*. Для установки файла нажмите клавишу i.

Некоторые файлы никогда не надо заменять с помощью mergemaster. К их числу относятся файлы, содержащие специфичную системную информацию, например */etc/passwd* или */etc/group*. Чтобы оставить их неизменными, нажмите клавишу d.

Нажатие клавиши m проведет через файл и позволит сопоставить отличия и объединить две версии. Это мощная возможность, ориентированная на тех, кто хорошо знаком с mergemaster, diff и содержимым */etc*, однако начинающим почти всегда лучше просто заменять или отклонять файлы.

Всю информацию о */etc* можно почерпнуть в главе 9.

Записи устройств

В главе 3 были кратко рассмотрены файлы устройств – файлы в каталоге */dev*, с помощью которых программы могут обмениваться данными с устройствами. Затем в главе 4 я описал ядро как интерфейс между аппаратным и программным обеспечением. При обсуждении процесса обновления эти понятия дополняют друг друга. Может сильно измениться как общение ядра с устройствами, так и интерфейс манипулирования этими устройствами. До сих пор в процессе обновления мы не изменяли специальные файлы в */dev*, однако если новое ядро обращается к старым файлам устройств, последствия будут непредсказуемы. Такие изменения происходят не при каждом обновлении, но необходимо знать, что это возможно.

---

[1] Такой формат называется «diff» (отличие). Он довольно распространен в мире UNIX.

В состав FreeBSD входит сценарий */dev/MAKEDEV*, создающий правильные файлы устройств. Предполагается, что сценарий */dev/MAKEDEV*, поставляемый с тем или иным ядром, будет создавать правильные файлы устройств именно для этого ядра. mergemaster сравнит сценарий */dev/MAKEDEV* из обновленного исходного кода и старый сценарий */dev/MAKEDEV*, а затем предложит установить новый. Сделайте это.[1] Правильные файлы устройств действительно необходимы. Если сценарий */dev/MAKEDEV* изменен, mergemaster предложит запустить его. Согласитесь и с этим. Повторю: правильные файлы устройств жизненно важны.

Последние шаги

После завершения работы mergemaster каждый обновленный файл в системе будет находиться на своем месте. Теперь достаточно перезагрузить систему, чтобы завершить обновление FreeBSD!

При обновлении системы может случиться что угодно. Выполнение команды `make buildworld` может быть не завершено, или система позднее может работать странно. Действия при возникновении каких-либо неполадок подобны шагам, предпринимаемым при неудачной сборке ядра. Поищите описание ошибки в архиве почтовых рассылок FreeBSD. Если ничего не найдено, пошлите последние пять-шесть строк вывода команды сборки по адресу *FreeBSD-questions@FreeBSD.org*. Включите в письмо следующую информацию:

- Последние строки вывода команды, инициировавшей неудачную компиляцию
- Номер версии FreeBSD
- Содержимое */var/run/dmesg/boot*
- Вывод `uname -a`

Упрощение процесса обновления через CVSup

Итак, процесс обновления нам понятен, и его можно упростить. Несколько изменений в */etc/make.conf* сведут часть процесса обновления, связанную с CVSup, к команде из двух слов. Сам я не сторонник такого подхода, но так поступают многие, поэтому далее описано, что следует предпринять.

Прежде всего, необходимо установить несколько переменных в */etc/make.conf*.

```
SUP_UPDATE=     yes
```

Предыдущая строка разрешает вызов CVSup командой make.

```
SUP=            /usr/local/bin/cvsup
```

Параметр `SUP` задает расположение CVSup по умолчанию. Если программа CVSup перемещена или надо указать полный путь к исполняемому файлу cvsup, это можно указать здесь.

---

[1] Если, конечно, вы не испытываете удовольствия от странного поведения системы, таинственных аварий и потери данных.

```
SUPFLAGS=       -g -L 2
```

Параметр SUPFLAGS задает стандартные ключи для команды cvsup. Чтобы она выполнялась «молча», измените это значение на -g -L 0.

```
SUPHOST=        cvsup13.FreeBSD.org
```

В этой строке приводится достаточно близкое зеркало FreeBSD с cvsup:

```
SUPFILE=        /usr/share/examples/cvsup/stable-supfile
```

Значение SUPFILE задает конфигурационный файл cvsup.

```
PORTSSUPFILE=   /usr/share/examples/cvsup/ports-supfile
```

PORTSSUPFILE указывает, какой supfile следует использовать для обновления «портов». Это значение надо определять лишь при необходимости обновить коллекцию «портов».

```
DOCSUPFILE=     /usr/share/examples/cvsup/doc-supfile
```

Наконец, DOCSUPFILE – это supfile для исходного кода коллекции документов. Оставьте это значение неопределенным, если не требуется обновлять дерево документации (в том числе Справочник, FAQ, статьи и т. д.).

Примечание

DOCSUPFILE обозначает не саму документацию, а ее исходный код. Если в системе нет инструментов для компоновки документации, этот код практически бесполезен.

После установки этих значений команду cvsup stable-supfile можно заменить на такую:

```
# make update
```

Некоторые администраторы находят, что более приятно просто перейти в */usr/src* и набрать make update && make buildworld && make install, нежели набирать полную команду CVSup. На ваше усмотрение.

Сборка локального сервера CVSup

Во многих организациях есть не одна система FreeBSD. Однако при обновлении из исходного кода каждый отдельный сервер должен соединяться с CVS-сервером FreeBSD и загружать новейший код. Это добавляет хлопот. Например, все зеркала поддерживаются добровольцами, которые предоставляют для этого свои серверы и полосу пропускания. Зачем загружать одно и то же снова и снова?

Кроме того, код, загружаемый на разные машины, может слегка отличаться, если они подключаются к различным серверам CVS. Предположим, вы вошли в систему на каждой машине и запустили CVSup. В промежутках между началом обновления исходного кода на разных машинах код на сервере CVSup может немного измениться. Код на зеркалах обновится не сразу, а процесс обновления на ваших машинах уже идет. Если у вас несколько ра-

бочих систем, то лучше всего, если они будут абсолютно идентичны. Даже если на них установлены версии -stable где-то между 4.4-release и 4.5-release, такое расхождение представляет собой потенциальный источник проблем. Его устранение сильно облегчит выявление неполадок. «Та-ак. Сервер 1 загибается. Может быть, из-за того что на серверах установлены различные версии FreeBSD?» Это дорога к безумию.

Преодолеть такие трудности можно с помощью запуска центрального сервера CVSup (известного еще как сервер «cvsupd»), который, по сути, является локальным зеркалом. Его обновлением можно управлять, и это гарантирует, что на всех машинах будет устанавливаться один и тот же код. Такой подход не только расположит к вам (или, по крайней мере, не оттолкнет от вас) администраторов, обслуживающих зеркала, но и защитит от проблем, которые могли возникнуть из-за разного кода на ваших машинах. Неполадки все еще будут возможны из-за различных настроек в */etc/make.conf*, но эти файлы можно сравнить и выяснить, в чем тут дело. Намного проще сравнить два файла, чем несколько тысяч!

Запустить сервер CVSup не так-то просто, но есть способ облегчить эту процедуру. «Порт» */usr/ports/net/cvsup-mirror* поддерживает все мудреные этапы конфигурирования зеркала. При установке «порта» программа cvsup-mirror задаст несколько вопросов; есть значения по умолчанию, но многие из них следует изменить. Установка программного обеспечения подробно рассмотрена в главе 9, однако установка этого «порта» довольно проста.

Прежде всего следует убедиться в наличии соединения с Интернетом, после чего введести следующие команды:

```
# cd /usr/ports/net/cvsup-mirror
# make install clean
```

На экране появятся сообщения, в том числе сообщения компилятора, которые уже должны распознаваться. (Возможно, вы не знаете, что они обозначают, но их надо отличать от других сообщений.)

В какой-то момент процесс установки приостановится и появится приглашение для ввода информации:

```
Master site for your updates [cvsup-master.FreeBSD.org]?
(Основной сайт для обновлений)
```

Сайт по умолчанию, *cvsup-master.FreeBSD.org*, зарезервирован для официальных зеркал FreeBSD; его могут использовать только владельцы официальных зеркал, которые предоставляют глобальный доступ (world access) к своим системам. Остальным следует задать один из восьмидесяти с лишним общедоступных серверов CVSup. При установке зеркала CVSup надо заранее узнать имя ближайшего общедоступного зеркала. Введите это имя.

Следующее приглашение будет выглядеть так:

```
How many hours between updates of your files [1]?
(Сколько часов должно проходить между обновлениями файлов?)
```

Сценарий внесет изменения в */etc/crontab* (см. главу 9), чтобы CVSup запускался автоматически. Можно принять значение по умолчанию или без труда его изменить. Если принять значение по умолчанию, то система будет обновлять саму себя ежечасно через cron. Так действуют официальные зеркала. Я обычно ввожу 168. В этом случае репозитарий обновляется раз в неделю, поскольку я не обновляю программное обеспечение на своих серверах чаще! Первое обновление займет немало времени, однако последующие обновления обычно занимают несколько минут.

Примечание

В большинстве случаев я обновляю сервер CVSup вручную, запуская сценарий */usr/local/etc/cvsup/update.sh*. Для обновления программ на группе машин до одной и той же версии -stable достаточно обновить код на сервере CVSup, а затем обновить программное обеспечение на всех машинах уже с этого сервера. Зачастую я обновляю код на одной машине, провожу многоэтапное всестороннее тестирование качества и затем обновляю программы на остальных машинах с того же сервера CVSup. Такой подход обеспечивает хорошее качество кода и идентичность систем. Нет никаких требований к частоте обновления кода; в конце концов, это ваш код, поэтому исходите из своих потребностей! Если сервер обновляется вручную, то надо убрать из */etc/crontab* записи, инициирующие автоматическое обновление! Файл */etc/crontab* рассмотрен в главе 9.

```
Do you wish to mirror the main source repository [y]?
(Зеркалировать репозитарий основного исходного кода?)
```

В большинстве случаев как раз нужен репозитарий основного исходного кода, поэтому значение по умолчанию обычно подходит.

```
Where would you like to put it [/home/ncvs]? /repo (Где его разместить?)
```

Здесь надо ввести путь к каталогу на диске, в котором будет размещаться зеркало. Часто я добавляю в систему отдельный маленький диск, предназначаю его для зеркала и называю */repo*. Вполне можно оставить и значение по умолчанию */home/ncvs*.

Поскольку вам, вероятно, нужен только репозитарий основного исходного кода, ответьте «n» на следующие три вопроса:

```
Do you wish to mirror the installed World Wide Web data [y]? n
(Зеркалировать установленные данные Сети?)
Do you wish to mirror the GNATS bug tracking database [y]? n
(Зеркалировать базу данных GNATS (для отслеживания ошибок)?)
Do you wish to mirror the mailing list archive [y]? n
(Зеркалировать архив почтовых рассылок?)
```

Конечно, если надо зеркалировать весь сайт *www.FreeBSD.org*, включая базу данных PR и архивы почтовых рассылок, ответьте «y». Однако помните: архивы почтовых рассылок огромны. На момент написания этих строк размер собственно репозитария исходного кода намного превышает 1 Гбайт, и он непрерывно увеличивается.

В ответах на следующие вопросы указывайте уникальные идентификаторы пользователя и группы. (Избегайте слов «nobody», «nonroot» или «no-

group».) Можно оставить значения по умолчанию или изменить имена пользователей и групп согласно локальной схеме:

```
Unique unprivileged user ID for running the client [cvsupin]?
(Уникальный ID непривилегированного пользователя для запуска клиента)
Unique unprivileged group ID for running the client [cvsupin]? ]?
(Уникальный ID непривилегированной группы для запуска клиента)
Unique unprivileged user ID for running the server [cvsup]? ]?
(Уникальный ID непривилегированного пользователя для запуска сервера)
Unique unprivileged group ID for running the server [cvsup]? ]?
(Уникальный ID непривилегированной группы для запуска сервера)
```

В завершение отмечу, что максимальное количество одновременных подключений клиентов легко изменить впоследствии, поэтому не волнуйтесь при ответе на следующий вопрос. Прекрасно подойдет значение по умолчанию:

```
Maximum simultaneous client connections [8]?
(Максимальное количество одновременных подключений клиентов)
```

После того как вы ответите на вопросы, процесс `make install` возобновится: будут добавлены имена пользователей и настроена конфигурация. Все готово для дальнейшей работы.

Управление доступом

Желание быть хорошим системным администратором и наличие личного репозитария не означает, что вы разрешаете всем желающим загружать что-либо с установленного зеркала CVSup. Сервер CVSup позволяет управлять доступом компьютеров к зеркалу.

Файл */usr/local/etc/cvsup/cvsupd.access* управляет списком хостов, которым разрешено подключаться к зеркалу. Строки, начинающиеся со знака «решетки» (#), являются комментариями, знак «плюс» (+) означает, что клиент может подключиться, а дефис (-) подразумевает, что такой возможности нет. Звездочка (*) означает, что клиент должен пройти аутентификацию (см. следующий раздел «Аутентификация»).

Каждое правило в *cvsupd.access* может ссылаться либо на имя хоста, либо на IP-адрес; IP-адрес предпочтителен. Также с IP-адресами можно применять сетевые маски.

Например, можно разрешить доступ из сети 192.168.0.0/16 и запретить доступ клиентов с любых других адресов такими строками:

```
+192.168.0.0/16
-0.0.0.0/0
```

Управление доступом по IP-адресам хорошо подходит для статических сетей. Например, поставщик услуг Интернета (Internet Service Provider, ISP) знает IP-адреса своих серверов и может легко указать их в *cvsupd.access*. Однако в других организациях, где клиенты подключаются со случайных IP-адресов, может потребоваться более гибкая система. Например, когда я занимался консультированием, к моему зеркалу был открыт доступ с любого

IP-адреса. Впрочем, для подключения пользователи должны были ввести свои имена и пароли. Если *cvsupd.access* пуст, доступ предоставляется при прохождении аутентификации по имени пользователя и паролю.

Аутентификация

Аутентификация применяется, когда необходимо разрешать подключения к зеркалу CVSup с любого адреса в Интернете. Для аутентификации сервер CVSup использует систему «оклик-отзыв» (challenge-response), а не передает пароли открытым текстом. Когда клиент подключается, он объединяет *общий ключ* (*shared secret*) – аналог «пароля» в CVSup – и системное время, а затем пропускает получившееся выражение через шифратор. Ту же операцию выполняет сервер. Теоретически клиент и сервер выполняют одни и те же вычисления с одними и теми же секретными данными и должны получить одинаковый ответ. Если зашифрованное сообщение клиента соответствует результату, полученному сервером, сервер предполагает, что клиент знает ключ, и разрешает ему доступ.

Это очень надежная система. Например, если кто-нибудь установит сетевой анализатор пакетов, он не добудет пароль. Более того, поскольку система «оклик-отзыв» применяет в вычислениях значения времени, перехваченный отзыв нельзя использовать повторно.

Для аутентификации нужен файл с паролями, */usr/local/etc/cvsup/cvsupd.passwd*. Права на его чтение должны быть только у пользователя CVSup, чтобы никто другой не смог получить информацию о пользователях. (Для обеспечения такой защиты надо запустить `chown cvsup cvsupd.passwd` и `chmod 600 cvsupd.passwd`.) Если файла с паролями нет, то управление доступом будет осуществляться только на основе файла *cvsupd.access*.

Пустые строки и строки с комментариями (начинаются с #) в *cvsupd.passwd* игнорируются. Первая строка в *cvsupd.passwd* – это имя сервера и секретный ключ, разделенные двоеточием.

```
magpire.AbsoluteBSD.com:testkey
```

Имя сервера отсылается назад клиенту, а секретный ключ используется для увеличения степени случайности. Секретный ключ может отсутствовать – система паролей CVS и без того обеспечивает случайность вычислений, но двоеточие, предшествующее секретному ключу, указать надо. Секретный ключ не может содержать двоеточие.

Следующие строки файла – легальные пользователи. Каждому пользователю отводится отдельная строка следующего формата:

```
user ID:shared secret:class:comment
```

Идентификаторы (ID) CVSup – это адреса электронной почты, например *mwlucas@AbsoluteBSD.com*. Общий ключ (shared secret) вычисляется на основании пароля пользователя. Поле класса (class field) зарезервировано на будущее; его надо оставить пустым. Наконец, поле комментариев (comment field) предназначено для администратора. Например, если кому-либо пре-

доставляется доступ к зеркалу CVSup, то здесь можно разместить комментарий, объясняющий причину разрешения доступа. (Вы это помните сейчас, но вспомните ли спустя год или два?)

Команда cvpasswd(1) автоматизирует создание записей в *cvsupd.passwd*. Это команда с двумя аргументами: адресом электронной почты пользователя и именем сервера. Пароль для этого пользователя будет запрошен дважды. Затем будут выданы несколько инструкций.

```
# cvpasswd mwlucas@AbsoluteBSD.com magpire.AbsoluteBSD.com
Enter password: (Введите пароль)
Enter same password again: (Введите тот же пароль еще раз)

Send this line to the server administrator at magpire.AbsoluteBSD.com: (Пошлите эту
строку администратору сервера magpire.AbsoluteBSD.com)
---------------------------------------------------------------------------------
❶ mwlucas@AbsoluteBSD.com:$md5$bf489b753a0a949a1c63a3f5da0d61b6::
---------------------------------------------------------------------------------
Be sure to send it using a secure channel! (Убедитесь, что передача данных пройдет
по безопасному каналу!)

Add this line to your file "$HOME/.cvsup/auth", replacing "XXX"
with the password you typed in: (Добавьте эту строку в файл "$HOME/.cvsup/auth",
заменив "XXX" на пароль, который вы ввели)
---------------------------------------------------------------------------------
magpire.AbsoluteBSD.com:mwlucas@AbsoluteBSD.com:XXX:
---------------------------------------------------------------------------------
Make sure the file is readable and writable only by you! (Убедитесь, что права на
чтение и запись этого файла есть только у вас!)
#
```

Зашифрованная строка в середине вывода (❶) содержит имя пользователя и общий ключ, основанный на пароле. Пошлите эту строку пользователю, которому разрешено подключение. Упомянутый «безопасный канал» (secure channel) означает, что данные, передаваемые по нему, не могут быть перехвачены злоумышленниками. Можно позвонить пользователю по телефону и прочитать ему код,[1] или записать его на дискету и доставить ее лично, или зашифровать код с помощью PGP и послать его по электронной почте. Если послать код в обычном, незашифрованном письме, то любой злоумышленник, перехвативший письмо на пути к адресату, сможет использовать код для доступа к серверу CVSup. Впрочем, если кто-либо похитил эту информацию, риск несанкционированного доступа невелик; пользователю по-прежнему нужен пароль для доступа к зеркалу.

Пользователь, получивший эту строку, помещает ее в файл *.cvsup/auth* в своем домашнем каталоге. Этот файл можно скопировать на любую систему, которую надо обновлять с сервера CVSup. Кроме того, необходимо, чтобы никто иной не мог читать этот файл. Для обеспечения такой защиты запустите `chmod 600 .cvsup/auth`.

---

[1] Хорошая идея! Телефон никогда не прослушивается, верно; и к нему никогда не подключаются соседи? – *Примеч. науч. ред.*

На машине-сервере скопируйте эту же строку в */usr/local/etc/cvsup/cvsupd.passwd*. Она соответствует записи passwd минимального формата. При желании в конец строки можно добавить комментарий.

Когда эти записи внесены в файлы *passwd* на машине-сервере и машине-клиенте, каждый раз при запуске CVSup и подключении к серверу у клиента будет запрашиваться пароль.

Примечание

Если на сервере нет ни *cvsupd.access*, ни *cvsupd.passwd*, любой желающий может подключиться к серверу с любого адреса в Интернете. Проект FreeBSD будет рад каждому, кто хочет организовать зеркало, но вы должны понимать, что делаете!

Совмещение аутентификации и доступа

Совмещение аутентификации и авторизации по IP-адресу представляет некоторые трудности. Если IP-адрес хоста представлен в списке допустимых адресов, у клиента не надо требовать пароль, а пользователю, имеющему пароль, нельзя отказывать в доступе, если подключение с его IP-адреса не разрешено. В конце файла *cvsupd.access* есть неявное правило «аутентификации». Если доступ клиента не заблокирован явным правилом «отказывать» (при подключении с того или иного IP-адреса), то клиенту будет позволено пройти аутентификацию. Никакого специального конфигурирования не требуется.

В примере файла *cvsupd.access*, представленном ранее, был явно запрещен доступ со всех IP-адресов, не указанных в списке. Если другим пользователям необходимо предоставить возможность проходить аутентификацию, в список надо внести IP-адреса, доступ с которых разрешен всегда, и отдельные блоки адресов, с которых никогда нельзя подключаться. Вот пример с комментариями:

```
#разрешить подключение любого пользователя внутри компании
+192.168.0.0/16
#разрешить подключение любого пользователя из компании-филиала
+10.10.0.0/16
# пользователям с этих адресов вход закрыт
-24.0.0.0/8
```

В этом примере клиенты с IP-адресами, начинающимися с 192.168 или 10.10, могут подключаться всегда. Компьютерам с IP-адресами, начинающимися с 24., в доступе отказано, даже если у пользователя есть имя и пароль. Если подключение производится с компьютера, чей IP-адрес не приведен в строках выше, у пользователя будет запрошено имя и пароль.

Такая схема обеспечивает полный контроль над доступом к зеркалу.

7

Организация защиты системы

В защищенной системе доступ к ресурсам компьютера открыт лишь авторизованным пользователям в дозволенных целях. Даже если в системе нет важных данных, большое значение имеют время процессора, память и полоса пропускания. Вообще, многие администраторы, считавшие свои системы маловажными и не заботившиеся об их защите, обнаруживали, что их машины невольно стали ретрансляторами (relays) – плацдармом для атак, наносящих урон целым корпорациям. Кому хочется проснуться утром от восхитительных звуков, издаваемых сотрудниками правоохранительных органов, которые ворвались в дом, потому что ваш незащищенный компьютер был использован для проникновения в банк.

Конечно, есть вещи похуже, чем подростки, захватившие власть над серверами, – скажем, перелом обеих ног. Однако вторая по значимости неприятность – однажды прийти на работу и обнаружить, что веб-страница компании теперь сообщает: «Ха-ха, здесь был Вася!»

К сожалению, за последние несколько лет стало намного проще взламывать удаленные компьютеры. Фирменные программы с «мышиным» интерфейсом, позволяющие громить компютеры, становятся общедоступными. Их можно найти, зайдя на нелегальные поисковые машины, подобные *Astalavista.com*. Сттит только одному способному взломщику написать программу, вскрывающую уязвимые места в защите, и тысячи скучающих подростков не находят ничего лучшего, как загрузить этот код и усложнить жизнь всем остальным. Даже если вас не заботит ваша система, ее надо защищать.

Вообще говоря, взламываются не операционные системы, а программы, работающие в них. Даже изначально и параноидально безопасная операционная система, самая надежная в мире,[1] не может защитить плохо написанные программы от них самих.

---

[1] Скорее всего, это OpenBSD. Или любая ОС на компьютере, который не подключен ни к одной сети, укрыт под 12 футами железобетона и по возможности разбит на миллиард крошечных частей, погруженных на несколько месяцев в водный раствор фтористого водорода.

Иногда проблемы, заложенные в таких программах, накладываются на особенности операционной системы и подвергают ее нешуточному риску. Чаще всего это выражается в *переполнении буферов,* когда программа злоумышленника попадает прямо в область памяти, из которой процессор берет команды для выполнения в данный момент, а операционная система выполняет этот код. Система FreeBSD прошла всестороннее тестирование и доработку, призванную исключить возможность переполнения буферов, однако нет никакой гарантии, что такая возможность полностью устранена. Новые функции и программы появляются каждый день, а их взаимодействие с устоявшимися функциями может быть непредсказуемым.

В этой главе внимание сосредоточено на наложении «заплаток» и обеспечении защиты систем. (Аудит сетевой архитектуры – это тема для нескольких толстых книг, и ее освещение не входит в задачу книги о FreeBSD.) FreeBSD предоставляет много инструментов для защиты системы от сетевых атак.

Кто враг?

Прежде всего, я произвольно распределю потенциальных взломщиков на три группы: малышей со скриптами, недовольных пользователей и опытных взломщиков. Более точную градацию взломщиков можно найти в книгах, посвященных безопасности, но такие подробности не относятся к теме данной книги. Эти категории легко объяснить, легко понять; под них подпадают 99% взломщиков, с которыми вам, вероятно, придется столкнуться.

Малыши со скриптами

Наиболее многочисленные взломщики – *малыши со скриптами* (*script kiddies*). Малыши со скриптами – это не системные администраторы. Они не искусны. Они загружают маленькие программы с «мышиным» интерфейсом, предназначенные для организации атак, и ищут потенциальные цели. Малыши со скриптами похожи на подростков с ружьями, разъезжающих на автомобилях в поисках легкой добычи. К счастью, от них защититься легче всего; для этого достаточно своевременно обновлять систему и программы на сервере, ставить на них заплатки.

Недовольные пользователи

Вторая группа, вызывающая большинство проблем безопасности, – это *ваши пользователи*. И правда, недовольные служащие пробивают большинство брешей в защите, потому что они как никто другой знают уязвимости в системе безопасности. Например, на всех ваших серверах своевременно устанавливаются заплатки, но в коммутационном шкафу есть модем, который позволяет любому желающему, который знает пароль, войти в сеть, обойдя брандмауэр. В таком случае ждите неприятностей.

Лучший способ остановить таких людей – не быть сентиментальным. Когда сотрудник покидает компанию, измените все пароли и сообщите всем слу-

жащим о его уходе и о том, что с ним больше не надо делиться информацией. Избавьтесь от незащищенного модема, недокументированного telnet-сервера и любого другого серьезного инструмента, в спешке размещенного там, где его якобы никто не сможет найти.

Опытные взломщики

Последняя группа действительно опасна: *опытные взломщики*. Это квалифицированные системные администраторы, исследователи систем безопасности и специалисты по преодолению защиты, желающие заполучить специфические данные о той или иной компании. Если один из них захочет проникнуть в систему, он, вероятно, это сделает.

Тем не менее надлежащие мероприятия по обеспечению защиты, которые остановят первые две группы злоумышленников, могут заставить опытных взломщиков изменить тактику. Вместо того чтобы проникать в компьютеры по сети, они будут вынуждены прикидываться ремонтниками телефонной компании и тайком устанавливать анализатор пакетов либо разгребать мусор в поисках листков с записанными паролями. В этом случае им придется быть у всех на виду, и они могут посчитать, что проникновение в систему не стоит возможных неприятностей.

Совет

Нередко можно услышать, как слово «хакер» употребляется для описания людей, которые взламывают компьютерные системы. Это слово имеет различные значения в зависимости от того, кто его произносит. В техническом мире хакером называют человека, который проявляет интерес к внутренней работе технических систем. Одни хакеры интересуются буквально всем, другие имеют более узкую область интересов, например компьютеры. В сообществе FreeBSD «хакер» – это знак уважения. Основной технический список рассылки называется FreeBSD-hackers. В средствах массовой информации хакерами называют взломщиков компьютерных систем, и точка. Для них все хакеры – негодяи. Я рекомендую во избежание путаницы вообще не употреблять это слово. В этой книге взломщики систем будут именоваться «злоумышленниками».[1] Технических кудесников можно величать по-разному, но они редко возражают против обращения «Sir» или «Madam».

Сообщения, относящиеся к безопасности FreeBSD

Лучший способ остановить всех взломщиков – своевременно обновлять систему. Другими словами, надо знать, когда обновлять систему и что именно обновлять. Устаревшие системы – это лучшие друзья малышей со скриптами.

В Проекте FreeBSD есть команда разработчиков, специализирующихся на аудите исходного кода и обеспечении безопасности как базовой операционной системы, так и добавляемого программного обеспечения. Эти разработчики поддерживают почтовую рассылку с небольшим трафиком, *FreeBSD-security-notifications@FreeBSD.org*, на которую стоит подписаться. Общие

[1] Лично я даю им гораздо менее приятные имена.

уведомления появляются в других почтовых рассылках (таких как BugTraq и CERT), рассылка security-notifications представляет собою единственный удобный источник информации, относящейся к FreeBSD.

Подписка

Для того чтобы подписаться на почтовую рассылку security-notifications, пошлите сообщение по адресу *majordomo@FreeBSD.org*. В письме должно быть следующее:

```
subscribe FreeBSD-security-notifications
```

В ответ придет сообщение с подтверждением. Где-то внутри него будет примерно такая строка с командами:

```
auth abax55b3 subscribe FreeBSD-security-notifications      mwlucas@AbsoluteBSD.com
```

Ответьте по адресу *majordomo@FreeBSD.org* письмом, содержащим эту строку, и процесс подписки завершен.

Чтобы отказаться от подписки, пошлите подобное сообщение по адресу *majordomo@FreeBSD.org*. В теле письма должен быть такой текст:

```
unsubscribe FreeBSD-security-notifications
```

В ответ придет сообщение со строкой подтверждения, которую надо послать обратно почтовому серверу. После этого подписка будет аннулирована.

Что вы получите

В почтовой рассылке попадаются два вида сообщений: советы по безопасности FreeBSD (*FreeBSD security advisories*) и советы по безопасности коллекции «портов» FreeBSD (*FreeBSD ports-collection security advisories*). У них совершенно разное назначение.

Советы по безопасности FreeBSD относятся к базовой операционной системе. Обнаружив «прореху» в том или ином компоненте FreeBSD, команда, ответственная за безопасность, публикует совет по безопасности. Прочтите его внимательно и определите, что надо предпринять.

Коллекция «портов» без преувеличения содержит тысячи программ, которые нетрудно установить на FreeBSD. Хотя это не полный список программ, которые могут работать в системе, это, безусловно, их значительная часть. Когда команда, обеспечивающая безопасность, находит «дыру» в одном из этих пакетов, она направляет уведомление производителю и публикует совет по безопасности коллекции «портов». Эти программы не подлежат контролю со стороны Проекта FreeBSD, однако, поскольку они поставляются с FreeBSD, в адрес разработчиков системы зачастую поступают упреки, что одна из этих программ взломана. Поэтому команда, обеспечивающая безопасность, выпускает такие советы для защиты пользователей. Если вы не установили программу, к которой относится тот или иной совет, беспокоиться не о чем.

Сообщения по безопасности обоих типов обычно содержат описание проблемы, исправления и обходные пути. Читайте советы внимательно – можете не сомневаться, что малыши со скриптами ищут уязвимые машины, которые можно взломать. Лучше всего быть к этому готовым.

Примечание

В этой главе будет рассмотрено много инструментов для обеспечения безопасности. Хотя ни один из них не является достаточным, все они желательны. Каждый способ, изученный в этой главе, следует воспринимать как один инструмент из целого комплекта, а не как панацею. Например, простое увеличение уровня безопасности не обеспечит полную защиту системы, но оно принесет пользу, если эту меру объединить с разумными правами доступа, флагами файлов, «заплатками», управлением паролями и всеми остальными шагами, которые подразумеваются хорошей политикой безопасности.

Установка профилей безопасности

При установке FreeBSD (версии 4.2 или одной из следующих) есть возможность выбора профиля безопасности, который, по сути, включает или отключает сетевые сервисы и задает настройки системы безопасности. Значения по умолчанию устанавливаются согласно значениям, предоставляемым Проектом FreeBSD. (Все изменения, вносимые профилем безопасности, производятся в */etc/rc.conf*.) В большинстве случаев эти профили являются отправной точкой, а конфигурацию, задаваемую профилем, надо отредактировать согласно вашим требованиям к ней. В следующих разделах в общих чертах описаны два профиля безопасности: moderate (обычный) и extreme (суровый).

Moderate

Профиль безопасности moderate включает inetd, sendmail и sshd. Таким образом, система может отправлять и получать электронную почту, а пользователи могут удаленно подключаться к системе через ssh. Кроме того, если в системе заблаговременно настроена поддержка NFS, то будет запущен portmap, а значит, система сможет предоставлять сервисы NFS. Уровень безопасности остается равным значению по умолчанию (-1).

Extreme

При выборе *профиля безопасности extreme* основные сетевые демоны системы не запущены, за исключением дополнительного, специально установленного программного обеспечения. Уровень безопасности равен 2. Система не будет получать или отправлять почту, и к ней нельзя подключиться удаленно. Она закрыта от взломщиков, поскольку не обменивается данными с внешним миром.

Профили безопасности предоставляют полезные шаблоны, но необходимо знать, как конфигурировать каждый из этих сервисов. Для этого следует обратиться к файлу *rc.conf* (см. главу 9).

root, группы и права доступа

Система безопасности UNIX считается в некотором смысле грубой, поскольку один суперпользователь, root, может делать все, что угодно. Другие пользователи – безропотные слуги, покорно носящие кандалы, в которые их заковал root. Хотя в этих словах есть доля правды, приличный администратор может комбинировать применение групп и прав доступа, решая почти все вопросы безопасности более гуманным способом.

Пароль root

Для некоторых действий нужен полный контроль над системой, в том числе возможность манипулировать ключевыми системными файлами, такими как файлы ядра, драйверов устройств и файлы системы аутентификации. Учетная запись root разработана как раз для выполнения таких действий.

Для использования пароля root надо либо войти в систему как root, либо, если вы – член группы *wheel*, выполнить команду смены пользователя (switch user) su(1). (Группы будут обсуждаться в следующем разделе.) Я рекомендую su; информация о пользователях, выполнивших su, заносится в протокол. Кроме того, su можно применять на удаленной системе. Выполнить эту команду очень просто:

```
# su
Password:
#
```

Проверьте текущее имя пользователя с помощью команды id(1):

```
# id
uid=0(root) gid=0(wheel) groups=0(wheel), 2(kmem), 3(sys), 4(tty), 5(operator),
20(staff), 31(guest)
#
```

Теперь вы владеете системой – да, она *в вашей власти*. Будьте внимательны при каждом нажатии клавиши; небрежность может превратить жесткий диск в неформатированный кусок металла – такой же чистый, каким он был изначально. Используйте пароль root расчетливо, ибо каждый, кто знает пароль root, может причинить непоправимый вред системе. Не давайте его никому без крайней необходимости!

Естественно, возникает вопрос: «А кому нужен доступ с правами root?» Для большинства действий по конфигурированию, обсуждаемых в этой книге, пароль root необходим. Но как только система настроена должным образом, применение пароля root можно значительно сократить и даже приостановить. Один из самых простых способов добиться этого состоит в том, чтобы правильно использовать группы.

Группы пользователей

В UNIX пользователи классифицируются по *группам*, в каждую из которых входят пользователи, выполняющие сходные административные функции.

В группу «www» могут входить редакторы веб-страниц, а в группу «email» – администраторы, управляющие почтовым сервером. Доступ к тем или иным файлам и каталогам можно разрешать для определенных групп. Основная информация о группах задается в файле */etc/group*.

Каждая строка в файле *group* содержит четыре поля, разделенных двоеточиями. Первое поле – имя группы. Имена групп могут быть совершенно произвольными. При желании ту или иную группу можно назвать «xyzzy». Впрочем, имена для групп лучше выбирать исходя из их предназначения. Сегодня вы помните, что группа xyzzy управляет системой электронной почты, но вспомните ли вы об этом через шесть месяцев? Имена групп должны что-то означать.

Второе поле содержит зашифрованный пароль группы. Пароли групп способствовали укоренению плохих, с точки зрения безопасности, привычек, поэтому в большинстве современных систем UNIX они не поддерживаются. Однако прежние программы написаны в расчете на присутствие этого поля в */etc/groups*, поэтому вместо того, чтобы оставить это поле пустым или удалить его, используйте звездочку (*) в качестве «заполнителя».

Третье поле содержит уникальный цифровой идентификатор группы (GID, group ID). Для идентификации групп в большинстве внутренних программ FreeBSD применяется именно GID, а не имена.

Последнее поле – список всех[1] пользователей, входящих в эту группу. Для внесения пользователя в группу достаточно добавить его имя в этот список. Имена пользователей разделяются запятыми.

Отредактировав */etc/group*, полезно убедиться, что файл не содержит ошибок. Чтобы перепроверить свою работу, запустите chkgrp(8). Файл */etc/group* будет досконально проверен, и если команда chkgrp ничего не «скажет», значит, вы все сделали правильно.

Главная группа

В файле *group* нет полного списка всех пользователей, входящих в каждую группу. Одновременно с новым пользователем создается группа, в которую он и вносится. Имя этой группы совпадает с именем пользователя. Это «главная группа» пользователя (primary group). Пользователь автоматически становится членом своей главной группы, что отражено в файле */etc/passwd* (см. главу 9).

Главные группы не представлены в */etc/group*. Единственная запись, свидетельствующая об их существовании, – это поле главной группы в */etc/passwd*. Возможно, это одна из досадных особенностей главных групп, однако

[1] На самом деле обязательно включать в этот список лишь тех пользователей, для которых эта группа – дополнительная. В файле */etc/passwd* в строке каждого пользователя есть номер его главной группы, а вот для включения в дополнительную группу следует вписать имя пользователя в соответствующую строку файла */etc/group*. – *Примеч. науч. ред.*

добавление в */etc/group* строк для каждого отдельного пользователя может затруднить управление этим файлом.

Например, когда добавляется пользователь «pbardaville», система создает группу «pbardaville» и заносит в нее пользователя с этим именем. Запись об этом появится только в */etc/passwd*.

Такой порядок может показаться сложным, однако надо просто запомнить, что в файле */etc/passwd* больше «козырей», чем в */etc/group*.

Интересные группы, создаваемые по умолчанию

В состав FreeBSD входит несколько групп, поставляемых по умолчанию. Большинство из них используются системой и не требуют к себе внимания системного администратора. Тем не менее здесь представлены самые полезные, интересные и необычные группы. Добавление собственных новых групп упрощает администрирование, однако группы, приведенные в этом списке, есть в каждой системе FreeBSD.

bin

Группа для основных программ.

daemon

Группа, используемая различными системными сервисами, например системой печати.

dialer

Группа пользователей, имеющих доступ к последовательным портам.

games

Группа для игровых программ и файлов.

kmem

Группа для программ, обращающихся к памяти ядра, например fstat(1), netstat(1) и т. д.

mail

Группа для программ, управляющих почтовыми операциями.

man

В современных системах BSD не применяется, но соответствует пользователю man.

news

Группа для программ, предоставляющих доступ к Usenet.

nobody

Группа для ID пользователя без привилегий.

nogroup

Группа без привилегий.

operator

Группа, имеющая доступ к накопителям; как правило, предназначена для резервного копирования.

staff

Группа для персонала системы (system staff).

tty

Группа для программ, которые могут выдавать результаты на терминалы, например wall(1).

wheel

Группа для пользователей, которым разрешено применять пароль root. Если пользователь знает пароль root, но не входит в группу wheel, он не может стать «пользователем root».

Групповые права доступа

Если той или иной группе назначить определенные права доступа, их унаследуют все члены этой группы. Права доступа к файлу также называются *кодом доступа*[1] (mode).

Согласно схеме прав доступа, принятой в UNIX, каждый файл имеет три набора прав доступа: для владельца, для группы и для всех остальных. Существующие права доступа к тому или иному файлу можно увидеть, запустив команду ls(1) с ключом `-l`:

```
# ls -l
total 29
-rwxr-xr--  1 mwlucas  admins    1188 Sep 14 09:35 file1
-rw-------  1 mwlucas  admins   27136 Sep 14 09:36 file2
drwxr-xr-x  2 mwlucas  admins     512 Sep 14 09:52 otherstuff
#
```

Первая строка (`total 29`) отображает количество дисковых блоков размером 512 байт, которые используются файлами. (В данном случае один *блок* соответствует половине килобайта, или одной двухтысячной части мегабайта.) В каталоге, представленном в листинге, есть два файла, `file1` и `file2`. Каждый из них описан в собственной строке, в которой приведена базовая информация о файле и правах доступа к нему. Права доступа к этим файлам указаны в начале каждой строки и представляют собой последовательности символов «r», «w» и «x», например `-rwxr-xr--`.

Права доступа определяют, как каждая группа может использовать тот или иной файл. Права могут быть трех типов: read (`r`, чтение), write (`w`, запись),

---

[1] По-английски права доступа принято называть либо mode (так принято в UNIX), либо permissions (в Windows). На русский язык это переводят как угодно: привилегии, разрешения, режим, код доступа, режим доступа и пр. По смыслу вернее «права доступа», и мы будем пользоваться этим термином. – *Примеч. науч. ред.*

и execute (x, выполнение). Право на чтение означает возможность просматривать или копировать файл. Право на запись позволяет его изменять или перезаписывать. Право на выполнение подразумевает возможность запускать файл как программу – все программы являются исполняемыми файлами. Любой дефис (-) означает, что у пользователя нет соответствующих прав на этот файл.

Последняя запись, otherstuff, – это каталог. Об этом свидетельствует буква «d» в начале строки с правами доступа. Права на каталог управляют доступом к каталогу точно так же, как права на файл управляют доступом к файлу.

После прав доступа представлено количество ссылок на этот файл. Ссылки рассмотрены в главе 13. Далее можно увидеть владелеца файла и группу, а за ними – количество байт в файле. Далее следуют дата и время последней модификации файла. В последнем поле находится реальное имя файла.

Права доступа в сочетании с владельцами и группами очень гибки. Например, несколько файлов можно поместить в группу,[1] названную www, а затем установить для нее права на чтение и запись этих файлов. Следовательно, каждый член группы www сможет их редактировать. Установка таких прав позволит веб-мастерам управлять веб-сайтами. При этом другие пользователи не смогут подменить веб-страницы, а в предоставлении прав root группе www в этом случае не возникло необходимости.

Строка прав доступа состоит из десяти символов. Первый символ указывает, является ли элемент группой. Другие девять символов разбиты на три группы по три символа, каждая из которых обозначает привилегии: в первой группе показаны права владельца файла, во второй – права группы на доступ к файлу, а в третьей – права всех остальных пользователей.

Первый символ в каждой группе означает чтение (read), второй – запись (write) и третий – выполнение (execute). Рассмотрим листинг:

```
-rwxr-xr--  1 mwlucas  admins    1188 Sep 14 09:35 file1
```

Первая группа из трех символов – rwx. Это значит, что владелец файла, mwlucas, может читать, записывать и исполнять файл. Вторая группа символов, r-x, говорит, что члены группы admins могут читать и исполнять этот файл, но не могут его записывать. Последняя группа, r--, сообщает, что любой пользователь системы может читать этот файл, но не может записывать и исполнять его.

Изменение прав доступа

Права на файл также называют *правами доступа*. Команда chmod(1), или «change mode» (изменить права доступа), позволяет любому пользователю, имеющему право на запись того или иного файла, изменить права доступа

[1] Файл в группу поместить нельзя, в группе могут быть только пользователи. Наверное, автор имел в виду их. – *Примеч. науч. ред.*

к этому файлу. chmod можно применять разными способами (полный список представлен на странице руководства), но здесь речь пойдет о самом распространенном способе, позволяющем изменять права доступа. Возможно, это не самый легкий метод, но он встречается чаще всего, и каждый системный администратор обязан его понимать.

Представление прав доступа в выводе ls кажется топорным. Их трудно произнести, трудно набрать и с ними трудно работать. Профессионалы UNIX обычно не могут с этим долго мириться, особенно если работу легко упростить.[1] Показанные выше права доступа необходимо уметь «читать», но chmod позволяет применять более короткую форму записи.

Код доступа в короткой форме задается числом из трех цифр (диапазон цифр – от 0 до7).[2] Первая цифра представляет права владельца, вторая – права группы, а третья – права всех остальных пользователей. (Это восьмеричная математика (по основанию 8); она подобна двоичной математике, которая описана в главе 5, посвященной работе в сети.) Цифра 4 означает «чтение» (read), 2 – «запись» (write) и 1 – «выполнение» (execute). Для задания прав доступа к файлу надо сложить соответствующие числа. Яснее ясного, не правда ли? Не беспокойтесь, будем двигаться медленно; если коды доступа уже понятны, можно пропустить пару абзацев.

Предположим, что владелец файла должен получить права на чтение, запись и выполнение файла, группа – на чтение и выполнение, а остальные – только на чтение. В этом случае строка прав доступа будет выглядеть так: `rwxr-xr--`.

Первая цифра в коде доступа получается из прав владельца – первые три символа `rwx` строки прав доступа. Чтению соответствует 4, записи – 2, выполнению – 1, что дает нам первую цифру кода доступа 4 + 2 + 1 = 7.

У группы есть права на чтение и выполнение. Чтение – это 4, а выполнение – 1; 4 + 1 = 5. Итак, вторая цифра кода доступа – 5.

Наконец, остальные пользователи могут только читать файл. Чтение – это 4, и 4 в результате, поэтому третья цифра кода доступа – 4. Для изменения прав доступа введите команду chmod:

```
# chmod 754 file1
# ls -l file1
-rwxr-xr--  1 mwlucas  admin   1188 Sep 14 09:35 file1
#
```

Сплошь и рядом права доступа описываются кодом доступа. Стоит поработать с ним какое-то время, как его применение входит в привычку. Подго-

---

[1] В UNIX «проще» часто означает «легче произнести и быстрее набрать, но намного сложнее понять».

[2] В особых обстоятельствах применяется код доступа из четырех цифр. Подробную информацию можно найти в chmod(1). Обычно код доступа из четырех цифр используется для файлов устройств и других специальных файлов.

товьте тестовый файл и поэкспериментируйте с его правами доступа, чтобы как следует их освоить.

Изменение принадлежности файла

Программа chown(1) позволяет изменить владельца файла, а программа chgrp(1) – изменить группу. Обе программы принимают два аргумента: имя пользователя[1] и имя файла.

В следующем листинге владельцем *file1* является mwlucas, а wheel – его группа:

```
# ls -l file1
-rwxrwxr--  1 mwlucas  wheel  1188 Sep 14 09:35 file1
#
```

Группу можно изменить с помощью следующей команды chgrp:

```
# chgrp dns file1
# ls -l file1
-rwxrwxr--  1 mwlucas  dns  1188 Sep 14 09:35 file1
#
```

Теперь файл находится в группе dns.[2]

С помощью chown можно изменять как владельца, так и группу. Так можно изменить владельца:

```
# ls -l file1
-rwxrwxr--  1 mwlucas  wheel  1188 Sep 14 09:35 file1
# chown bind file1
# ls -l file1
-rwxrwxr--  1 bind     wheel  1188 Sep 14 09:35 file1
#
```

Для изменения владельца и группы надо разделить двоеточием имена, задаваемые в chown:

```
# chown bind:wheel file2
#
```

Примечание

Распределять права на файлы может только root. Если обычный пользователь хочет, чтобы его файлами владел кто-то другой, он не сможет выполнить `chown otheruser filename`. Точно так же, если пользователь не входит в ту или иную группу, он не сможет назначить ее какому-либо файлу.

---

1 В случае chgrp – имя группы. – *Примеч. перев.*

2 Автор смешивает два понятия – «находиться в группе» и «принадлежать группе». Членом группы может быть только пользователь. То, что в поле группы в свойствах файла указано имя группы dns, означает, что каждый член группы dns имеет такие права доступа к этому файлу, которые у свойств этого файла содержатся в поле прав доступа группы. – *Примеч. науч. ред.*

Назначение прав доступа

Итак, теперь ясно, как назначать права доступа и изменять владельца файла и его группу. Но что именно следует назначать или изменять?

Возьмем файлы, необходимые для жизненно важных системных ресурсов, например файлы зон сервера DNS (см. главу 11). Многие системные администраторы назначают владельцем этих файлов пользователя root. Ему же предоставляется исключительное право записи этих файлов. Таким образом, обычные пользователи не имеют к ним доступа.

Такой подход приемлем, если администратор один, но не годится при делегировании задач. Некоторые администраторы обходят возникающие трудности с помощью дополнительных программ, таких как sudo(8) (в */usr/ports/security/sudo*), однако при их конфигурировании легко ошибиться.

В прошлом у меня были помощники, в то время еще не вполне квалифицированные системные администраторы. В их задачу входило редактирование важных файлов, однако ни при каких обстоятельствах им нельзя было доверять пароль root. Решением явилось применение групп, что позволило мне разрешить администраторам доступ лишь к этим файлам, не наделяя их правами root. (В данном примере я буду использовать DNS, однако этот подход применим к любой системе, где требуется рарешить ограниченной группе пользователей редактировать набор файлов.)

Вначале рассмотрим, какой тип доступа надо предоставить пользователям. В примере с DNS владелец файлов должен иметь права на их чтение и запись, и членам группы надо назначить те же права. Всем остальным пользователям требуется установить права на просмотр файлов, но запретить их редактирование. Поскольку файлы DNS – это простые текстовые файлы, а не программы, то ни у кого не должно быть прав исполнять их. (Права на выполнение файла, который не является программой, не причинят вреда, но могут сбить с толку.) Поэтому строка прав будет выглядеть так: `rw-rw-r--`. Права владельца включают в себя чтение (4) и запись (2), члены группы могут читать (4) и записывать (2) файлы, а остальные пользователи имеют права только на чтение (4). Значит, права на эти файлы можно назначить командой `chmod 664 filename`.

Теперь надо назначить владельца файлов. При этом необходимо помнить, что многие системные программы запускаются от имени определенного пользователя. Например, упомянутый сервер DNS запускается от имени bind, а веб-сервер Apache – от имени nobody. Хотя по логике пользователь, от имени которого запущен сервер, является владельцем сервера, это не всегда так. Если кто-либо взломает сервер DNS, он может выполнять команды как пользователь bind. Можно не беспокоиться о том, что кто-либо прочтет эти файлы, но у неавторизованных пользователей не должно быть прав изменять их. Самое простое решение – создать отдельного пользователя, который будет владельцем этих файлов.

Создание нового пользователя

Нового пользователя можно создать с помощью adduser(8). (В главе 9 будут обсуждены adduser(8) и некоторые возможности */etc/login.conf*, которые сделают невозможной регистрацию в системе под именем этого пользователя.) Для полного отключения паролей обратитесь к программе vipw(8) (программа vipw(8) также будет рассмотрена в главе 9), а затем измените группу выбранного файла на «dns». Затем установите права на чтение и запись для владельца и группы, а для остальных – только на чтение:

```
# chown dns:dns file1
# chmod 664 file1
# ls -l file1
-rw-rw-r--  1 root  dns  1188 Sep 14 09:35 file1
#
```

Теперь сотрудники[1] могут выполнять свои задачи без пароля root, а ваши файлы будут защищены от системного процесса, использующего эти файлы.

Флаги файлов

Права доступа файловой системы UNIX – это стандарт для различных версий UNIX. Система BSD расширяет схему прав, вводя флаги файлов. Вместе с правами доступа эти флаги усиливают защиту системы. Некоторые флаги используются для функций, которые не связаны с безопасностью, однако здесь будут рассматриваться флаги, имеющие к ней отношение.

Примечание

Многие флаги оказывают различное влияние в зависимости от уровня безопасности системы, кратко рассмотренного в разделе «Уровни безопасности». Пока же кивайте головой и улыбайтесь при упоминании об этих уровнях; все станет ясно через несколько страниц.

Ниже представлены флаги, относящиеся к безопасности:

sappnd

Системный флаг «только добавление», который может устанавливать только root. В файл с этим флагом можно добавлять записи, однако удалять или редактировать этот файл нельзя (это особенно удобно для файлов протоколов). Если система каким-то образом «скомпрометирована», то установка флага sappnd файла *.history* может принести пользу. Тактика типичного злоумышленника состоит в том, чтобы удалить *.history* или сделать из него символическую ссылку на */dev/null*, чтобы администратор не узнал о предпринятых действиях. Благодаря установке флага sappnd малыши со скриптами не смогут замести следы таким образом. Забавно шпионить за кем-либо, кто пытается удалить файл с флагом sappnd. Можно уви-

[1] Не забудьте включить ответственных сотрудников в группу dns, как указано выше, изменив */etc/group*. – *Примеч. науч. ред.*

деть, как растет разочарование взломщика с каждой новой попыткой. (Конечно, лучше, если в систему вообще нельзя пробраться!) Этот флаг нельзя изменить, когда система работает на уровне безопасности 1 и выше.

schg

Системный флаг «неизменяемости», который может устанавливать только root. Файлы с установленным флагом `schg` нельзя изменять вообще: редактировать, перемещать, заменять. Сама файловая система пресекает все попытки коснуться этого файла каким-либо образом. Этот флаг нельзя изменить, когда система работает на уровне безопасности 1 и выше.

sunlnk

Системный флаг «запрет на удаление», который может устанавливать только root. Файл можно редактировать или изменять, но нельзя удалить. Этот флаг не настолько безопасен, как предыдущие два: если файл можно редактировать, то его можно очистить. Впрочем, это удобно при определенных обстоятельствах. Я использовал эту возможность, когда программа настаивала на удалении собственного файла протокола в случае аварии. Обычно установка любых стандартных флагов не приносит пользы. Этот флаг нельзя изменить, когда система работает на уровне безопасности 1 и выше.

uappnd

Пользовательский флаг «только добавление», который может устанавливать только владелец файла или root. Как в случае с установленным системным флагом `sappnd`, в файл с флагом `uappnd` можно добавляеть записи, однако этот файл нельзя удалять или редактировать. Такая возможность наиболее полезна для протоколов, которые ведут личные программы пользователя. Установка такого файла предохраняет пользователей от их собственных ошибок. Владелец файла и root могут удалить этот флаг в любое время.

uchg

Пользовательский флаг «неизменяемости», который может устанавливать только владелец файла или root. Как в случае с флагом `schg`, описанным ранее, флаг «неизменяемости» защищает пользователя от изменения файла. Как и в предыдущих случаях, root может преодолеть действие[1] этого флага. Кроме того, сам пользователь может убрать его, на каком бы уровне безопасности ни работала система. Этот флаг помогает предотвратить ошибки, но не обеспечивает безопасность системы.

uunlnk

Пользовательский флаг «запрет на удаление», который может установить только владелец файла или root. Файл с установленным флагом `uulink` владелец удалить не может, хотя root может его аннулировать, и этот флаг может быть снят. Флаг `uulink` обычно бесполезен, однако,

---

[1] И снять тоже может. – *Примеч. науч. ред.*

подобно другим пользовательским флагам, он может пригодиться для предотвращения ошибок.

Просмотр флагов файла

Флаги файла можно увидеть с помощью команды `ls -lo`:

```
# ls -lo important
-rw-r--r--  1 mwlucas  mwlucas  uchg 0 May 11 19:51 important
```

Символы `uchg` здесь говорят о том, что флаг «неизменяемости» установлен. Для сравнения приведем строку в случае, когда у файла не установлены флаги:

```
# ls -lo unimportant
-rw-r--r--  1 mwlucas  mwlucas  - 0 May 11 19:52 unimportant
#
```

Дефис вместо имени флага говорит о том, что флаги файловой системы не были установлены.

В свежеустановленных системах FreeBSD лишь немногие файлы отмечены таким образом. Впрочем, флаги можно установить как угодно. На одной системе, считавшейся взломанной, я неистово запускал `chflags -R schg` в различных системных каталогах, чтобы не позволить никому заменить системные исполняемые файлы на троянские версии. Это не остановило бы взломщика, но я почувствовал себя лучше, представив, как бы он был расстроен, если бы добрался до командной строки.

Установка флагов

Флаги можно устанавливать командой chflags(1). Например, чтобы защитить ядро от подмены, можно сделать следующее:

```
# chflags schg /kernel
```

Теперь никто не заменит ядро: ни злоумышленник, ни вы.

С помощью ключа `-R` можно рекурсивно изменять флаги, продвигаясь по дереву каталогов. Например, чтобы сделать каталог */bin* неизменяемым, воспользуйтесь такой командой:

```
# chflags -R schg /bin
```

Ура! Базовые исполняемые файлы теперь нельзя изменить.

Для снятия флага надо в команде chflags предварить имя флага словом «no». Например, чтобы снять флаг `schg`, только что установленный на ядро, введите следующую команду:

```
# chflags noschg /kernel
```

Уже говорилось о том, что снятие этого флага подразумевает работу системы на уровне безопасности -1. Поэтому, не откладывая в долгий ящик, обсудим уровни безопасности.

Уровни безопасности

Уровни безопасности – это установочные параметры ядра, которые изменяют поведение базовой системы, запрещая те или иные действия. С повышением уроня безопасности ядро будет вести себя немного иначе. Например, на низких уровнях безопасности флаги файлов, рассмотренные выше, могут быть сняты. Например, флаг «не удалять» можно снять, а затем удалить файл. На более высоких уровнях безопасности флаг не может быть снят. Подобные изменения имеют место в других частях системы. В целом, изменение поведения системы, вызванное повышением уровня безопасности, либо сорвет планы злоумышленника, либо остановит его. Уровень безопасности системы, вступающий в силу при начальной загрузке, можно задать с помощью параметров `kern_securelevel_enable="YES"` файла *rc.conf*.

Уровни безопасности усложняют эксплуатацию системы, поскольку накладывают определенные ограничения на ее поведение. В конце концов, многие операции, выполняемые при обычном администрировании, может применить и злоумышленник, заметающий следы. Например, при использовании уровней безопасности в системе необходимо установить дополнительные заплатки. Введение уровней безопасности позволяет расстроить козни среднего злоумышленника, намеревавшегося уничтожить данные, внедрить троянскую программу или причинить вред системе иным способом.

Установка уровней безопасности

Существует пять уровней безопасности: -1, 0, 1, 2 и 3, где -1 является низшим уровнем, а 3 – наивысшим.

Как уже говорилось, разрешить использование уровней безопасности можно с помощью установки `kern_securelevel_enable="YES"` в файле *rc.conf*. После этого уровень безопасности можно автоматически задавать при загрузке, указав его в переменной `kern_securelevel=X` файла *rc.conf*. Когда бы вы ни увеличили уровень безопасности системы, его нельзя снизить без перезагрузки в однопользовательском режиме. (Если бы снизить уровень можно было без перезагрузки, этим воспользовался бы злоумышленник.)

Уровень безопасности -1

Уровень безопасности -1, устанавливаемый по умолчанию, не подразумевает применения дополнительных средств защиты, предоставляемых ядром. Если вы изучаете и часто изменяете конфигурацию, оставьте уровень безопасности -1 и применяйте права доступа к файлам, встроенные в BSD, а также другие меры безопасности UNIX, достаточные в большинстве ситуаций.

Уровень безопасности 0

Уровень безопасности 0 используется только в начале загрузки системы. Он не предлагает никаких специальных функций. Когда система переходит в многопользовательский режим, уровень безопасности автоматически увеличивается до 1. (Установка `kern_securelevel=0` в */etc/rc.conf* эквивалентна уста-

новке `setting kern_securelevel=1`.) Значит, для использования уровня безопасности 0 не так много причин.

Уровень безопасность 1

На уровне безопасности 1 ситуация интереснее:

- Нельзя загружать и выгружать модули ядра с помощью kldload/kldunload (см. главу 4).
- Программы не могут записывать данные напрямую в системную память через устройства */dev/mem* или */dev/kmem*.
- На монтированные диски нельзя записывать данные напрямую, а значит, нельзя форматировать разделы. (Файлы можно записывать на диск через стандартный интерфейс ядра, нельзя лишь форматировать диски или адресоваться к диску, как к физическому устройству.)
- Нельзя запускать систему X Window.

Самое наглядное следствие уровня безопасности 1 в том, что нельзя изменять флаги файловой системы, специфичные для BSD. Если файл помечен как «неизменяемый», то заменить его не удастся.

Уровень безопасности 2

Уровень безопасности 2 предоставляет все преимущества уровня безопасности 1 с двумя добавлениями:

- Нельзя записывать данные напрямую в монтированные и немонтированные файловые системы.
- Системное время за раз можно изменить не более чем на 1 секунду.

Начинающий системный администратор может посчитать эти особенности несущественными, однако это важные приемы обеспечения безопасности. Хотя UNIX предоставляет удобные инструменты, например текстовые редакторы для записи данных в файл, без них можно обойтись. Точно так же можно обойтись и без файловой системы и получить прямой доступ к нулям и единицам, которые записаны на диске. В таком случае можно изменить любой файл вне зависимости от прав доступа. На практике такая возможность применяется лишь при установке нового жесткого диска. Обычно напрямую записывать данные на диск может только пользователь root. При уровне безопасности 2 это не позволяется даже ему.

Другой хакерский прием состоит в том, чтобы изменить системное время, отредактировать файл и установить прежнее время. Когда администратор ищет файлы, которые могли вызвать неполадки, искаженный файл выглядит так, словно к нему годами не притрагивались, а значит, не привлечет к себе пристального внимания.

Уровень безопасности 3

Уровень безопасности 3 называют сетевым режимом безопасности. Он проявляет себя точно так же, как уровень безопасности 2, но не допускает изме-

нений правил IPFW или IPFilter. (Эти программы обсуждаются в главах 11 и 8 соответственно.) Если в системе включена фильтрация пакетов или управление полосой пропускания, а их правила хорошо настроены и вряд ли будут изменяться, то можно установить уровень безопасности 3.

Как выбрать уровень безопасности?

Уровень безопасности, необходимый для той или иной операционной среды, зависит от ситуации. Например, если машина FreeBSD только что введена в эксплуатацию и ее требуется точно настроить, следует оставить уровень безопасности -1. По окончании настройки уровень безопасности можно увеличить. Для большинства систем вполне подходит уровень безопасности 2.

Для дополнительной защиты полезно установить флаги `schg` и `sappnd` на существенные файлы, поскольку добавленная защита от изменения времени предоставляет еще больше способов заставить хакера обнаружить себя.

Если применяется одна из программ FreeBSD, предназначенных для фильтрации пакетов и организации брандмауэра, то сттит задуматься об уровне безопасности 3. Однако, выбирая уровень 3, следует отдавать себе отчет в предпринимаемых действиях, иначе ждите неприятностей. Например, если система FreeBSD используется в качестве корпоративного брандмауэра, уровень безопасности 3 не позволит изменять конфигурацию брандмауэра без разрыва соединения с Интернетом. Как уже говорилось ранее, если уровень безопасности 3 установлен для ограничения доступа к определенным портам веб-сервера, то скорее всего такой уровень будет в самый раз.

Когда уровни безопасности и флаги файлов не помогают?

Рассмотрим случай, когда взломщик находит «дыру» в сценарии CGI на веб-сервере Apache, получает через него доступ к командной оболочке (shell), а затем с его помощью получает права root.

Поскольку уровень безопасности выбран правильно, планы взломщика, вероятно, сорвутся, так как он не сможет заменить ядро системы на другое, специально скомпилированное ядро. Тем не менее он сможет заменить множество системных исполняемых файлов на «троянских коней» с тем, чтобы при следующем входе в систему ваш пароль был отправлен на анонимный почтовый ящик с веб-интерфейсом или в сетевую телеконференцию.

Итак, для защиты своих ключевых файлов вы запускаете `chflags schg -R /bin/*`, `schg -R /usr/lib` и т. п. Отлично. Если вы забыли про один файл, скажем, какой-нибудь непонятный */etc/rc.i386*, то хакер сможет отредактировать его, добавив `chflags -R noschg /`. Позднее он может перезагрузить систему – ночью, когда вы этого не заметите. (Как часто вы садитесь и основательно проверяете файлы */etc/rc*?)

Вы думаете, что все файлы полностью защищены и система в безопасности. А что у вас в каталоге */usr/local/etc/rc.d*, где находятся локальные програм-

мы, запускаемые при начальной загрузке? В процессе загрузки система будет выполнять все файлы с расширением *.sh*, найденные в этом каталоге. Значит, гипотетический хакер может причинить немало вреда, разместив здесь простой сценарий командной оболочки. В конце концов, */etc/rc* увеличивает уровень безопасности в последнюю очередь, когда все программы уже запущены. А что если хакер создаст сценарий, который убьет запущенный */etc/rc* перед тем, как он увеличит уровень безопасности, а затем запустит свой собственный */var/.hidden/rc.rootkit*, чтобы завершить подключение к сети?

Конечно, это только один путь, есть и другие. Важно помнить, что организация защиты системы – это нелегкая задача, для которой нет одного простого решения. Как только взломщик получил доступ к командной строке, начинается схватка. И, если он умеет хоть что-то, о его появлении вы узнаете слишком поздно. Конечно, всегда лучше не впускать его в свой дом, чем бегать за ним по коридорам.

Жизнь с уровнями безопасности

Если вы достаточно свободно обращались с флагом `schg`, установка новой версии системы (или даже «заплатки») может оказаться неудобной. На самом деле те условия, которые усложняют жизнь хакерам, могут сделать вашу жизнь невыносимой, если не знать обходных путей. Итак, каковы же обходные пути?

Если файл */etc/rc.conf* защищен флагом `schg`, то для редактирования системных файлов сначала надо понизить уровень безопасности. Конечно, параметр, задающий уровень безопасности, находится в */etc/rc.conf*, поэтому для редактирования этого файла необходимо получить контроль над системой до запуска */etc/rc*.

Для этого загрузите систему в однопользовательском режиме и смонтируйте задействованные файловые системы. Поскольку на этом этапе уровень безопасности еще не установлен, вы можете смонтировать файловые системы, запустить `chflags noschg` на задействованных файлах и продолжить загрузку. Можно даже отредактировать */etc/rc.conf*, отключить уровни безопасности и лишь затем возобновить обычную загрузку. (В этом случае процесс пройдет быстрее, но защита с помощью флагов файлов будет потеряна.)

После внесения необходимых изменений уровень безопасности можно увеличить (но не уменьшить), не прибегая к перезагрузке. Для этого запустите команду sysctl:

```
# sysctl -w kern.securelevel=[desired securelevel]
```

Теперь, когда вы можете управлять изменениями файлов, рассмотрим управление доступом к системе из сети.

Программы, которые можно взломать

Как я уже упоминал в начале этой главы, обычно взламывается не операционная система, а программы, работающие в ней. Самая большая мишень –

это сетевые программы, поэтому возникает вопрос: как определить, какие программы являются сетевыми? Ответ: с помощью sockstat(1).

Sockstat(1), инструмент FreeBSD с дружелюбным интерфейсом, определяет, какие сокеты открыты в системе и какие программы «прослушивают» эти сокеты. Он показывает как устанавливаемые соединения, так и соединения, которые уже доступны для организации взаимодействия.

Сокет – это всего лишь «логическое устройство», которое «прислушивается» к соединению. Есть сокеты, которые «слушают» сеть; это сетевые порты, обсужденные в главе 5. Сокеты могут «прослушивать» сети IP версии 6; это сокеты IPv6. Наконец, есть сокеты, которые «вслушиваются» в локальный компьютер. Программы могут создавать сокеты для взаимодействия друг с другом.

Если у вас нет сети IPv6, то не надо заботиться и о сокетах IPv6. Кроме того, сокеты UNIX не расположены в сети; чтобы взаимодействовать с сокетом UNIX, необходимо войти в систему и пройти все испытания, предусматриваемые стандартными правами доступа UNIX. Если взломщик сделает это, у вас уже будут неприятности. Итак, рассмотрим открытые сетевые (Ipv4) сокеты, запустив `sockstat -4`. Вот снимок с экрана моего лэптопа:

```
# sockstat -4
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN ADDRESS
mwlucas  ssh        372    3 tcp4   192.168.1.200:1025    208.63.178.18:22
root     X          347    0 tcp4   *:6000                *:*
root     snmpd      296    6 udp4   *:161                 *:*
root     lpd        234    6 tcp4   *:515                 *:*
root     syslogd    209    4 udp4   *:514                 *:*

#
```

Каждая строка здесь представляет либо один открытый сокет, ожидающий подключения, либо установленное соединение. (Похоже на `netstat -na`, не правда ли?) Большинство колонок самоочевидны. `USER` – это пользователь, запустивший процесс; `COMMAND` – имя команды; `PID` – идентификатор процесса (process ID), удерживающего этот сокет. Имя команды очень полезно при организации защиты системы. `FD` – это файловый дескриптор процесса; сейчас его можно не принимать во внимание. Колонка `PROTO` – это протокол Интернета, использующий соединение. Наконец, колонки `LOCAL ADDRESS` и `FOREIGN ADDRESS` показывают IP-адреса и номера портов на каждой стороне соединения. Если в колонках `LOCAL ADDRESS` и `FOREIGN ADDRESS` есть IP-адреса и номера портов, значит, этой строкой представлено существующее соединение. Если в колонке `FOREIGN ADDRESS` показаны две звездочки, разделенные двоеточием (*:*), значит, эта программа «прислушивается» к входящим подключениям.

В предыдущем примере я использую ssh для подключения к удаленной системе с моего лэптопа. На самом деле ssh не «прислушивается» к входящим подключениям; sockstat показывает только мое соединение с другим сервером. В лэптопе к входящим подключениям «прислушиваются» четыре сервиса: X «прислушивается» к поключениям со всех локальных IP-адресов на

порту 6000/tcp, демон snmp запущен на порту 161/udp, а lpd и syslogd «прислушиваются» к входящим подключениям.

Вот важный момент. Каждый открытый сетевой порт является потенциальным слабым местом и целью взломщика. Остановите ненужные сетевые сервисы и обеспечьте защиту необходимых. Все ясно? Хорошо.

Примечание

Регулярно проверяйте, какие порты открыты в системе, ибо это поможет получить неожиданную информацию. Например, я установил net-snmp, чтобы получить snmpwalk и другие команды, имеющие к ней отношение. При этом я совершенно забыл, что уже установил демон snmp, который надо было остановить и не запускать вновь при начальной загрузке.[1]

Изучить вывод sockstat на лэптопе довольно просто, однако вывод sockstat на сервере Интернета – это совсем другое дело. На маленьком сервере вывод sockstat может состоять из сотен строк. Например, вот листинг для очень маленького сервера:

```
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN ADDRESS
root     sshd1    28971    5 tcp    192.168.15.18.22      24.2.72.241.35886
❷ nobody   httpd    27356   17 tcp    *.80                  *.*
nobody   httpd    27355   17 tcp    *.80                  *.*
nobody   httpd    27354   17 tcp    *.80                  *.*
nobody   httpd    27353   17 tcp    *.80                  *.*
nobody   httpd    27352   17 tcp    *.80                  *.*
nobody   httpd    27351   17 tcp    *.80                  *.*
root     named    72871    4 udp    *.2151                *.*
root     named    72871   20 udp    192.168.15.18.53      *.*
root     named    72871   21 tcp    192.168.15.18.53      *.*
root     named    72871   22 udp    127.0.0.1.53          *.*
root     named    72871   23 tcp    127.0.0.1.53          *.*
root     httpd    65199   17 tcp    *.80                  *.*
❸ root     sshd1      275    3 tcp    *.22                  *.*
root     sshd1      269    3 tcp    192.168.15.19.80      *.*
root     sendmail   214    4 tcp    *.25                  *.*
root     inetd      207    4 tcp    *.106                 *.*
❶ root     inetd      207    5 tcp    *.110                 *.*
root     inetd      207    6 tcp    *.113                 *.*
```

Назначение одних портов очевидно, других – нет. Так, вы, вероятно распознали httpd и sendmail. Но для чего предназначены все эти открытые порты inetd?

Чтобы получить ответ, выполните команду grep, задав в качестве аргументов номер порта и файл */etc/services* (см. главу 5). В выводе этой команды можно увидеть, какой сервис использует этот порт. Например, сервис, запущенный на порту 110, довольно странный (❶). grep ищет строки, соответст-

[1] «Что такое SNMP?» Я слышу ваш крик. Обратитесь к главе 19. «А как его остановить?» Обратитесь к главе 11.

вующие шаблону. В данном случае будут найдены все строки, содержащие последовательность символов «110». Для порта 110 grep выдаст следующие строки:

```
# grep 110 /etc/services
pop3            110/tcp    #Post Office Protocol - Version 3
pop3            110/udp    #Post Office Protocol - Version 3
nfsd-status     1110/tcp   #Cluster status info
nfsd-keepalive  1110/udp   #Client status info
softcm          6110/tcp   #HP SoftBench CM
softcm          6110/udp   #HP SoftBench CM
#
```

Команда grep возвращает несколько строк, в которых есть «110», однако пропустим те из них, которые к делу не относятся. Например, третья и четвертая строки содержат символы «110», но это ссылки на порт 1110. В первых двух строках представлен сервис pop3, рассмотренный в главе 12. Быстрый поиск по архивам почтовых рассылок FreeBSD показывает, что pop3 доставляет почту клиентам, таким как Eudora и Outlook.

Примечание

Чтобы выяснить, что собой представляет тот или иной сервис, надо поискать информацию о нем в Сети или отключить его и посмотреть, что при этом произойдет. Я успешно применял оба метода, однако в конечном итоге исследование предпочтительнее.

Этот метод можно использовать для идентификации других сервисов, предоставляемых демоном inetd. (Сам inetd обсуждается в главе 12.) В листинге sockstat также есть несколько экземпляров httpd (❷), веб-сервера. Один экземпляр демона ssh прослушивает порт 22 (❸), а другой – порт 80.[1] Также есть несколько записей named, прослушивающего порт 53, и одна запись named на порту 2151. Все эти программы будут обсуждаться позднее, когда разговор пойдет о сетевых сервисах, которые предоставляет FreeBSD. Сейчас важно понимать, что каждая серверная программа «прослушивает» сетевой сокет, а идентифицировать эти программы можно с помощью sockstat(1).

Итак, теперь ясно, какие сервисы запущены. Как отключить ненужные? Лучший способ закрыть эти порты – не запускать программы, которые их задействуют. Обычно сетевые демоны запускаются из двух мест: либо из

[1] Различные брандмауэры можно обойти, если послать трафик в обход порта 80, порта TCP, используемого для веб-трафика. При наличии интернет-сервера за пределами такого брандмауэра можно запустить sshd на порту 80 этого сервера и установить соединение из-за пределов брандмауэра. И хотя брандмауэр, полагающийся исключительно на то, что он ограничивает доступ к порту, не может помешать вам установить соединение таким способом, сам факт наличия такого брандмауэра должен недвусмысленно подсказать, что блокируемые сервисы не предназначены для того, чтобы вы их использовали. Как бы вас не уволили за такую сообразительность.

/etc/rc.conf, либо из сценария запуска в */usr/local/etc/rc.d*. Программы, интегрированные в основную систему FreeBSD, такие как sendmail, ssh и portmap, имеют флаги в */etc/rc.conf*, с помощью которых запуск этих программ можно разрешать или запрещать (см. главу 9). Дополнительные программы, такие как веб-серверы, запускаются из сценариев в */usr/local/etc/rc.d* (см. главу 11). Однако программа inetd – особый случай, поскольку ее назначение в том, чтобы запускать маленькие, редко используемые программы. Хотя запуск inetd как целого разрешается с помощью флага в *rc.conf*, программы, относящиеся к inetd, необходимо запускать и останавливать из inetd. Разрешение и запрещение запуска программ inetd описано в главе 12.

Для удаления ненужных сетевых сервисов запустите `sockstat -4` и идентифицируйте каждый процесс. Определите необходимые сервисы, отметьте их и отключите остальные; затем перезагрузите систему, чтобы убедиться в том, что изменения возымели эффект. Если результаты sockstat, запущенного после перезагрузки, вас удовлетворили, то работа сделана. В противном случае возвращайтесь к началу.

Собрать все вместе

Когда в системе открыты только необходимые порты и ясно, какие программы их используют, то о защите этих программ и следует позаботиться. Если разработчики, обеспечивающие безопасность FreeBSD, отправляют извещение о неполадках сервиса, который вы не применяете, это сообщение можно спокойно проигнорировать. Если же речь идет о «дыре» в используемой программе, на это необходимо обратить внимание. Подобная практика защитит от большинства малышей со скриптами. Такие инструменты, как флаги файлов или уровни безопасности, позволят минимизировать ущерб, который могут причинить взломщики, если они проникнут в систему. Наконец, использование групп для ограничения действий самих системных администраторов может защитить компьютеры от случайного или преднамеренного причинения вреда.

Далее речь пойдет о более эффективных инструментах для обеспечения безопасности, которые предлагает FreeBSD.

8

Расширенные средства защиты

В состав FreeBSD входит множество инструментов, обеспечивающих защиту сетевого трафика и пользователей. Например, средства управления трафиком позволяют запретить установку соединений с определенными сегментами Интернета. Реализовать это можно разными способами. Вы также можете засадить пользователей в виртуальную машину, называемую *клеткой* (*jail*), где они имеют доступ ко всему, кроме основного сервера. Помимо этих методов, в данной главе будут рассмотрены мониторинг защиты системы и шаги, которые необходимо предпринять потенциальной жертве злоумышленника.

Начнем с основ – с управления сетевым трафиком.

Управление трафиком

Системный администратор должен уметь управлять входящим и исходящим трафиком, чтобы не подпускать к системе незваных гостей. FreeBSD предоставляет множество инструментов, позволяющих управлять доступом к системе, осуществляемым из внешнего мира. Здесь внимание будет сосредоточено на TCP wrappers и фильтрации пакетов. Функции этих инструментов управления доступом значительно перекрывают друг друга, поэтому они составляют идеальную пару.

Программа TCP Wrappers управляет доступом к особым серверным программам (также известным как *демоны*). Запросы на установление соединений передаются программе TCP Wrappers, которая оценивает запросы согласно своей конфигурации. Конфигурировать TCP Wrappers довольно просто; для этого не нужны глубокие знания сетей, однако серверные программы должны быть собраны с учетом работы с TCP Wrappers.

Средства фильтрации пакетов регулируют, какие пакеты принимает система. Отклоненный запрос на соединение не доходит до программ, запущенных на сервере; он отклоняется на низком уровне сетевого стека. Средства фильтрации пакетов могут управлять трафиком к любой программе, сервису или сетевому порту, но требуют хорошего знания сетей.

В любом случае перед реализацией управления трафиком необходимо выбрать одну из двух стратегий: *принимать все пакеты по умолчанию* (*accept*) или *отвергать их* (*deny*).

Принимать по умолчанию или отвергать?

Один из важнейших вопросов в любой системе защиты – принимать ли все пакеты по умолчанию или же отвергать их. Принятие по умолчанию означает разрешение соединений любого типа, за исключением тех, которые запрещены явным образом. Отвергать по умолчанию, значит, разрешать соединения лишь с указанных адресов Интернета, а все другие соединения запретить. Выбрав стратегию по умолчанию, можно регулировать доступ к системе, ограждая или открывая те или иные сервисы.

На самом деле выбор между принятием или запрещением по умолчанию – это выбор между предоставлением сервисов всему миру или же немногим избранным. Также следует учитывать, возможен ли доступ к системе для всех желающих.

Если система представляет собой корпоративный веб-сервер, то можно сделать его видимым только для пользователей корпоративной сети. Такая стратегия означает **запрет** по умолчанию и явное указание тех, кто может устанавливать соединение с системой. (По мере возможности я выбираю именно этот подход.) Альтернативный подход – открыть систему для всех, за исключением нескольких лиц. Такая стратегия означает **принятие** по умолчанию.

Следует отметить, что выбор той или иной стратегии не означает, что все сервисы в системе должны следовать установке по умолчанию. При конфигурировании машины, подключенной к Интернету, я выбираю стратегию запрета по умолчанию и явным образом разрешаю всеобщий доступ к веб-серверу. Запросы на установление соединения с другими программами, расположенными на этой машине, отклоняются, если они не исходят с нескольких специально указанных IP-адресов. Это совершенно приемлемая стратегия запрета по умолчанию.

На стратегии принятия и запрещения по умолчанию я буду ссылаться в следующих разделах.

TCP Wrappers

В главе 5 говорилось, что сетевые соединения устанавливаются различными программами, ожидающими запросов на соединение. TCP Wrappers перехватывает эти запросы до того, как они достигнут демона, анализирует IP-адрес, сверяясь с конфигурационным файлом, и принимает соответствующее решение: принимать, отвергать или изменять тот или иной запрос. Вопреки своему названию, TCP Wrappers работает как с сетевыми соединениями UDP, так и с соединениями TCP. TCP Wrappers – это установившийся стандарт UNIX, который был внедрен в FreeBSD. Тем не менее отдельные

программы могут и не работать с TCP Wrappers. Почти все программы, составляющие основную часть FreeBSD, работают с TCP Wrappers, однако некоторые программы сторонних разработчиков – нет.

Наиболее часто Wrappers применяются для защиты inetd – программы, которая запускает меньшие демоны. (Программа inetd обсуждается в главе 12.) Для запуска inetd с поддержкой wrappers применяйте флаг `-Ww` – например, `inetd_flags="-Ww"` в */etc/rc.conf* (см. главу 9). Примеры, представленные здесь, работают лишь при правильном запуске inetd. Хотя в этих примерах TCP Wrappers будут применяться для защиты программ inetd, этим же способом можно защитить любую другую программу.

Конфигурирование Wrappers

TCP Wrappers проверяют каждый входящий запрос на соединение, последовательно сверяясь с правилами в */etc/hosts.allow*. Как только найдено первое совпавшее правило, процесс немедленно прекращается. Значит, порядок правил очень важен.

Каждое правило занимает отдельную строку. Она состоит из трех частей, разделенных двоеточиями: имени демона, списка клиентов и списка параметров. Вот простой пример такой строки:

```
ftpd : all : deny
```

В этом примере имя демона – «ftpd», а список клиентов – «all», что означает все хосты. Наконец, параметр «deny» означает «отвергать все соединения». Никто не сможет подключиться к серверу FTP на этом хосте, если более раннее правило не предоставило такой доступ.

В предыдущих примерах мы ссылались только на два параметра: `accept` и `deny`. Соответственно, они *разрешают* или *запрещают* соединения. Есть много других параметров, которые будут рассмотрены позже.

Имя демона

Имя демона – это название программы, набираемое в командной строке. Например, inetd запускает программу ftpd, когда получает входящий FTP-запрос. Веб-сервер Apache запускает программу, называемую httpd. Значит, если ваша версия Apache поддерживает wrappers, «httpd» можно указывать в */etc/hosts.allow*. Специальное имя демона, ALL, соответствует всем демонам, которые поддерживают wrappers.

Если за сетевой картой закреплено несколько IP-адресов, можно указать разные правила для каждого IP-адреса, за которые отвечает демон. В этом случае IP-адрес представляет собой часть имени демона:

```
ftpd@192.168.8.7 : ALL : deny
ftpd@192.168.8.8 : ALL : accept
```

В этом примере есть два имени демона: ftpd@192.168.8.7 и ftpd@192.168.8.8. Для каждого из них есть отдельное правило TCP Wrappers.

Список клиентов

Список клиентов – это список разделенных пробелами конкретных IP-адресов, блоков сетевых адресов, имен хостов, имен доменов и ключевых слов. Имена хостов и IP-адреса просты, и их надо лишь перечислить:

```
ALL: netmanager.AbsoluteBSD.com 192.168.4.3 : allow
```

Номер сети в списке клиентов представляет собой IP-адрес и сетевую маску, разделенные косой чертой (см. главу 5). Например, если малыши со скриптами атакуют с группы различных адресов, которые начинаются с 216.136.204, их можно блокировать так:

```
ALL: 216.136.204.0/255.255.255.0 : deny
```

В шаблоне клиента можно также указывать имена доменов, предваряя их точкой:

```
ALL : .mycompany.com : allow
```

Конечно, можно задавать и обратные действия – запрещать соединения с какого-то одного домена:

```
telnetd : .competitor.com : deny
```

Если список клиентов достаточно длинный, то его можно хранить в файле и прописывать к нему путь, в поле клиента в */etc/hosts.allow*. Исходя из моего опыта, в сети ISP возможно большое количество хостов с произвольно разбросанными адресами, а в корпоративной сетевой среде рабочие станции, управляющие сетью, могут быть разбросаны по всему миру. Каждая рабочая станция, как и любая другая рабочая станция, использовала одно и то же правило TCP Wrappers и занимала полдюжины строк в */etc/hosts.allow*. Поддерживая список всех рабочих станций в одном файле, я смог централизовать все изменения (редактирование одного файла позволяет обновить все правила, которые вызывают этот файл).

Клиентские ключевые слова

Помимо явно указанных адресов и имен клиентов можно использовать несколько специальных ключевых слов, позволяющих добавить в список группу клиентов:

ALL

Под это ключевое слово подпадает любой возможный хост.

LOCAL

Под это ключевое слово подпадает любой хост, чье имя не содержит точку. Обычно это машины в локальном домене.

UNKNOWN

Под это слово подпадают машины с неопределимыми именами хостов, IP-адресами или именами пользователей. Как правило, если машина устанавливает соединение IP, ее IP-адрес известен. Однако для определения

имен хостов необходим DNS, а для отслеживания имен пользователей – протокол identd. Этот параметр следует использовать очень внимательно, так как временные неполадки в DNS могут сделать неразрешимыми даже имена локальных хостов, а в большинстве хостов протокол identd не работает по умолчанию. Никто не хочет, чтобы машина стала недоступной только потому, что сервер имен неверно настроен, особенно если эта машина и есть сервер имен!

KNOWN

Под это ключевое слово подпадает любой хост с определимым именем и IP-адресом. Повторюсь: пока сервер DNS недоступен,[1] не удастся узнать имя каждого хоста в Интернете. Если вы говорите, что к машине могут подключаться все опознаваемые хосты, а сервер DNS недоступен, то подключение не будет позволено никому.

PARANOID

Под это ключевое слово подпадает любой хост, имя которого не соответствует его IP-адресу. Представим запрос на соединение с хоста с адресом 192.168.84.3, который выдает себя за *mail.AbsoluteBSD.com*. Получив запрос, TCP Wrappers проверит, какой IP-адрес у *mail.AbsoluteBSD.com*. Если TCP Wrappers получит IP-адрес, отличающийся от исходного IP-адреса, то хост подпадает под это правило.

Большинству ключевых слов, характеризующих подключающегося клиента, приведенных здесь, необходим работающий сервер DNS (см. главу 12). Применяя эти ключевые слова, следует помнить о существенной связи между DNS и другими программами. Если DNS не работает, то демоны, использующие wrappers и ключевые слова, не смогут распознать ни один хост. Это значит, что все хосты подпадут под правила `UNKNOWN`. Кроме того, недействующий DNS на стороне удаленных пользователей может сорвать их доступ к вашим серверам, поскольку ваши серверы DNS не смогут получить необходимую информацию с клиентских серверов DNS.

Другие ключевые слова доступны, но не так полезны или безопасны. Например, разрешать соединения можно на основе имени пользователя на удаленной машине, от которой исходит запрос. Однако так поступать не стоит. Любой желающий может добраться до компьютера с FreeBSD или Linux и назвать себя так, как ему вздумается. Если настройки TCP Wrappers на моей домашней системе разрешают подключение лишь пользователю с именем «mwlucas», то при желании кто-нибудь легко добавит учетную запись с этим именем на своей машине FreeBSD. Кроме того, доступ по имени пользователя опирается на ранее упомянутый протокол identd, поддерживаемый небольшим количеством хостов. На странице руководства hosts_access(5) можно найти другие малоизвестные параметры – той же полезности.

---

[1] Здесь и везде в документации «служба (сервис) недоступна(ен)» означает, что от соответствующей службы нельзя получить ответ в ожидаемый срок. Это может означать перегрузку в сети, зависание сервера, зависание конкретной службы, например сервера имен, разрыв связи и много другое. – *Примеч. науч. ред.*

Ключевые слова ALL и ALL EXCEPT

Ключевые слова ALL и ALL EXCEPT можно использовать как для имени демона, так и для списка клиентов.

Под ключевое слово ALL подпадает абсолютно все. Например, по умолчанию файл */etc/hosts.allow* начинается с правила, разрешающего все подключения, с любых адресов и к любому демону:

```
ALL   : ALL: accept
```

Доступ разрешен ко всем программам от всех клиентов. Его можно ограничить, если конкретизировать список клиентов или список демонов:

```
ALL   : 192.168.1.87 : deny
```

В этом примере отвергаются все подключения с хоста 192.168.1.87.

Категорически заблокировать доступ от всех хостов – не самая лучшая идея. Однако следует помнить, что программа TCP Wrappers читает правила последовательно и останавливается на первом подходящем правиле. Ключевое слово ALL позволяет легко установить запрет или разрешение по умолчанию. Рассмотрим следующий набор правил:

```
ALL   : 192.168.8.3 192.168.8.4 : accept
ftpd  : ALL : accept
ALL   : ALL : deny
```

В этом примере рабочим станциям 192.168.8.3 и 192.168.8.4 разрешен доступ ко всем демонам. Это настольные машины системного администратора. Далее, любой желающий может подключиться к сервису FTP на этой машине. Наконец, все остальные соединения запрещены. Это удобно при выборе стратегии запрета по умолчанию.

Ключевое слово ALL EXCEPT применяется для еще большего сокращения предыдущего набора правил. ALL EXCEPT позволяет перечислять хосты как исключения; под правило подпадает то, что не перечислено в списке. Рассмотрим тот же набор правил, переписанный с применением ALL EXCEPT:

```
ALL   : 192.168.8.3 192.168.8.4 : accept
ALL EXCEPT ftpd : ALL : deny
```

Одни администраторы считают более понятными правила с ALL, другие – с ALL EXCEPT. Важно помнить, что первое подходящее правило останавливает сличение, поэтому применение правила ALL требует осторожности. Вообще говоря, первое правило, содержащее любую комбинацию ALL и ALL EXCEPT в списке демонов и списке клиентов, останавливает сличение; под это правило подпадает любое соединение.

Параметры allow

Параметр *allow* предписывает TCP Wrappers принять соединение. По умолчанию файл *hosts.allow* начинается с такого правила:

```
ALL : ALL : allow
```

Это правило применимо для всех демонов и всех клиентов; под него подпадают и, соответственно, разрешаются все возможные соединения. Если необходимо защитить сервисы, то это правило не должно быть первым в списке. В то же время оно хорошо подойдет в качестве последнего правила, если требуется лишь защитить определенные серверные программы от подключений с тех или иных сетевых адресов.

При экспериментировании с TCP wrappers полезно разрешить любые соединения с локального хоста, иначе многие программы откажутся работать, пока не смогут общаться с локальной машиной. Разрешить эти соединения можно так:

```
ALL : localhost : allow
```

Параметры для ответных действий

Теперь, прояснив понятия списка демонов и списка клиентов, рассмотрим параметры для ответных действий, представляющие больший интерес. Суть этих параметров очень проста. Есть входящий запрос на установление соединения, подпадающий под то или иное правило. Что с ним делать? Ответные действия могут быть очень простыми или очень сложными и хитроумными.

Примечание

Если использовать много параметров, то правила TCP Wrappers могут быть очень длинными. К счастью, в файле *hosts.allow* можно применять обратную косую черту (\ ) с последующим переходом строки (Return). Это символ продолжения строки, благодаря которому правила удобочитаемы.

Самые главные параметры – это *accept* и *deny*. Если запрос на соединение подпадает под то или иное правило, запрос либо передается ожидающему демону, либо отклоняется. Однако можно применять и дополнительные параметры, разделяя их двоеточиями.

severity

Как только сделан выбор между принятием и отклонением запроса на подключение, информацию о запросе можно *зарегистрировать* в протоколе. Предположим, вы хотите блокировать все запросы, исходящие от конкурента; может оказаться полезным знать, предпринимал ли он попытки установить соединение. Об этом могут рассказать файлы протоколов.[1] Точно так же может потребоваться знать о том, сколько запросов на соединение было отклонено от машин с неполадками в DNS (особенно если указано ключевое слово PARANOID).

[1] Если необходимо протоколировать все попытки установить соединение через любой порт, то более надежный подход заключается в использовании sysctls: net.inet.tcp.log_in_vain и net.inet.udp.log_in_vain (см. приложение). Эти sysctls протоколируют все неудачные попытки обращения к любому порту системы, а не только к защищаемым демонам.

Параметр *severity* (строгость) посылает сообщение программе системного протоколирования (system log), syslogd(8). Программу syslogd можно настроить на переадресацию сообщений в произвольный файл (см. главу 19), в зависимости от источника (facility) syslogd и выбранного уровня важности сообщения:

```
telnetd: ALL: severity auth.info : allow
```

Это правило подразумевает протоколирование всех соединений telnet.

twist

Параметр *twist* позволяет выполнять произвольные команды оболочки (shell) и сценарии, когда кто-либо пытается установить соединение с защищаемым демоном TCP, и возвращает вывод команд пользователю. twist работает только с соединениями TCP.[1] twist принимает команду оболочки как аргумент и действует по правилу «отклонить и что-нибудь сделать». Для использования twist необходимо знать основы создания сценариев командной оболочки; возможны очень сложные варианты применения twist, но здесь будут представлены более простые. В конце концов, здесь не демонстрируются сценарии! При возникновении сомнений всегда можно запустить /bin/echo "*причина*", чтобы объяснить удаленному клиенту причину отклонения запроса на соединение. (Обратите внимание на прямые двойные кавычки вокруг причины – они важны!)

Если используется запрет по умолчанию, то twist полезен для последнего правила. (Если защита выстроена по стратегии запрещения, добавьте в конец правил безопасности правило «поймать все и запретить».) Параметр twist можно использовать для возврата ответа лицу, пытающемуся установить соединение. Например, так:

```
ALL : ALL : twist /bin/echo      "Вы не можете использовать этот сервис."
```

Для того чтобы не предоставлять конкретный сервис тому или иному хосту, можно явно указать имя демона и клиента. Запись в следующем примере немного не вмещается в строку, поэтому я разбил ее на части с помощью символа обратной косой черты:

```
sendmail : .spammer.com : twist /bin/echo \
           "Вы не можете использовать этот сервис"
```

Если вы настроены дружелюбно, можете оповещать людей о причине отклонения их запросов на соединения. Следующее правило с twist отклоняет все запросы на установление соединения, исходящие от хостов, чьи имена не соответствуют их IP-адресам, и объясняет причину отказа:

---

[1] Строго говоря, это не так. Однако вспомните из главы 5, что UDP – это протокол без установления соединения; не существует соединения, по которому можно вернуть ответ. Значит, необходимо предпринять изощренные и трудоемкие усилия, чтобы twist мог работать с UDP. Кроме того, те программы, которые используют UDP, обычно не ожидают подобного ответа и не готовы его принять и интерпретировать. Применение twist для UDP не стоит возникающих трудностей.

```
ALL : PARANOID : twist /bin/echo \
      "Ваш DNS неработоспособен.  Возвращайтесь, когда устраните неполадки."
```

twist удерживает сетевое соединение открытым, пока команда оболочки не выполнена. Если она выполняется долго, система будет удерживать больше открытых соединений, чем планировалось. Это может значительно снизить производительность системы. Правила с twist должны быть просты и выполняться быстро.

Примечание

Может показаться заманчивым разместить в выводе twist грубое сообщение, особенно если, по вашему мнению, ни у кого не может быть законных оснований для доступа к серверу. Однако выпад «На-ка, выкуси!», направленный в адрес малышей со скриптами, разозлит легитимных пользователей и еще больше раззадорит малышей.

spawn

Подобно twist, параметр *spawn* отклоняет запрос на соединение и запускает указанную команду оболочки. В отличие от twist, spawn не возвращает результаты клиенту. Параметр spawn следует применять, когда при получении запроса на соединение система FreeBSD должна выполнить те или иные команды, о которых не должен знать клиент. Эти команды выполняются в фоновом режиме, а их результаты не возвращаются клиенту. Правило из следующего примера разрешает соединение, но заносит IP-адрес клиента в файл протокола:

```
ALL : PARANOID : spawn (/bin/echo %a >> /var/log/misconfigured) \
      : allow
```

Читатель, знакомый с языком оболочки, призадумается над символом %a в предыдущей команде. Программа TCP Wrappers поддерживает множество переменных, которые можно применять в командах twist и spawn. Эти переменные раскрываются перед выполнением команд, поэтому с их помощью можно легко видоизменять ответы на запросы. Переменная %a означает адрес клиента. Будучи размещенной в команде, она преобразуется в IP-адрес до выполнения этой команды. Другие переменные представлены в табл. 8.1.

Таблица 8.1. Переменные, которые можно использовать в командах twist и spawn

Переменная	Описание
%a	Адрес клиента
%A	IP-адрес сервера
%c	Вся доступная информация о клиенте
%d	Имя демона
%h	Имя хоста клиента (если доступно) или IP-адрес
%H	Имя хоста сервера (если доступно) или IP-адрес

Таблица 8.1 (продолжение)

Переменная	Описание
%n	Имя хоста клиента; если оно не найдено, выдается UNKNOWN. Если имя хоста не соответствует IP-адресу, это эквивалентно PARANOID
%N	Имя хоста сервера; если оно не найдено, возвращается либо UNKNOWN, либо PARANOID

Эти переменные можно применять везде, где предоставляемая ими инфомация используется в сценариях оболочки. Например, чтобы занести всю доступную информацию о клиенте в файл протокола при каждом подключении к защищаемой программе, можно использовать spawn:

```
ALL : PARANOID : spawn (/bin/echo %c >> /var/log/clients) \
      : allow
```

Можно заметить, что этот сценарий похож на сценарий из предыдущего примера. Незначительные изменения коснулись переменной и имени файла протокола. То же самое можно сделать с любой информацией, которую надо протоколировать.

Символы пробела и обратной косой черты в имени хоста могут вызвать нарекания у командной оболочки, поскольку это недопустимые символы. При обычных обстоятельствах они не должны присутствовать в командах, однако кто-нибудь может вставить пробел в имя хоста, чтобы сбить с толку программы защиты. Для обеспечения безопасности программа TCP Wrappers замещает все символы, которые могут вызвать сложности у командной оболочки, на символ подчеркивания (_). Проверьте наличие таких символов в файлах протоколов; они могут свидетельствовать о попытках проникновения в систему или о том, что кому-то нравится символ подчеркивания в именах хостов.

Собрать все вместе

Возьмем все примеры, уже рассмотренные в этой главе, и создадим файл */etc/host.allow* целиком, чтобы защитить гипотетическую систему, подключенную к сети. Для начала составим перечень сетевых ресурсов, предлагаемых системой, а также список IP-адресов, закрепленных за системой, и пользователей, которым разрешено подключение:

- Диапазон IP-адресов – 192.168.0.0/16. Система предлагает telnet, ftpd и portmap(8).
- Диапазон IP-адресов конкурента, которому запрещен доступ к системе,[1] такой: 10.5.4.0/23.

[1] Специально отметим, что блокировать доступ конкурента к сервисам, предоставляемым для всех остальных, – это не лучшая идея. Сотрудники конкурирующей фирмы могут легко получить к ним доступ, установив коммутируемые соединения. При этом вы будете выглядеть не лучшим образом.

- Примем отчасти параноидное решение: хосты с неверной информацией DNS могут принадлежать взломщикам, поэтому запросы на соединение, исходящие от этих машин, будут отклоняться.[1]
- Хосты в локальной сети могут использовать демон portmap, но хостам из других сетей это запрещено. С любого адреса Интернета можно запрашивать доступ к серверам FTP и telnet. (Конечно, для доступа к любому сервису по-прежнему нужны имя пользователя и пароль!)

Хотя эти требования довольно сложны, они сводятся к очень простому набору правил:

```
#отклонять все запросы на соединение, исходящие от конкурента и от хостов
#с неверными записями DNS
ALL : PARANOID 10.5.4.0/23 : deny
#разрешить использовать portmap в локальной сети
#и запретить для всех остальных.
portmap : ALL EXCEPT 192.168.0.0/16 : deny
#теперь, когда portmap защищен и конкурент блокирован, разрешим telnet и FTP
ALL : ALL : allow
```

Примеры с более подробными комментариями можно найти в файле */etc/hosts.allow* или на странице руководства hosts_allow(5).

Фильтрация пакетов

Многие дополнительные программы, доступные в коллекции портов FreeBSD (FreeBSD Ports Collection, см. главу 10) не могут использовать TCP Wrappers. Для этих программ можно применять IPFW или IPFilter – инструменты фильтрации пакетов, работающие на уровне ядра.

При фильтрации пакетов каждый сетевой пакет, поступающий в систему, сверяется со списком правил, которые предписывают ядру, что надо сделать с пакетом. Когда соответствующее правило найдено, ядро действует исходя из этого правила. Например, согласно тому или иному правилу, фильтр может пропустить, удалить или видоизменить пакет. Однако здесь нельзя применять замечательные параметры, разрешенные в TCP Wrappers; клиенту не уходит сравнительно дружелюбное «сообщение о запрете» – вместо этого соединение разрывается на сетевом уровне.

Идея фильтрации пакетов достаточно проста, но первая реализация этой стратегии может стать нелегким испытанием. Будьте готовы потратить несколько часов на эксперименты с фильтрацией пакетов и не отчаивайтесь при неудачах. Даже если вы думаете, что понимаете работу IP, единственный способ по-настоящему изучить этот протокол состоит в том, чтобы работать с ним.

---

[1] Это очень осторожная стратегия. Хосты с неверными записями DNS обычно расположены в сетях, в которых серверам DNS не уделяется внимание или же их обслуживают некомпетентные/переутомленные администраторы. Однако на всех хостах, с которых атакуют взломщики, информация DNS скорее всего искажена.

Примечание

Для эффективной фильтрации пакетов необходимы солидные знания работы IP. Попытки организовать фильтрацию без понимания ее сути будут тщетными и бессмысленными. При наличии сомнений перечитайте кажущийся бесполезным материал главы 5.

IPFilter

IPFilter – это модуль фильтрации трафика, который здесь будет рассмотрен подробно. Возможно, это самая сложная система фильтрации пакетов, доступная в любых открытых и закрытых программах. IPFilter разработан независимо от FreeBSD, но был интегрирован в основную ОС несколько лет назад. (Также он работает в Solaris и в других версиях BSD.)

Для применения IPFilter сначала необходимо пересобрать ядро (см. главу 4) и включить следующие параметры:

```
options         IPFILTER
options         IPFILTER_LOG
```

Параметр `IPFILTER` добавляет в ядро базовую поддержку IPFilter; действия IPFilter протоколируются благодаря `IPFILTER_LOG`. Хотя для работы IPFilter в протоколировании нет жесткой необходимости, модуль протоколирования довольно мал, поэтому его тоже можно включить. (Иначе рано или поздно придется пересобирать ядро, чтобы включить протоколирование для устранения тех или иных неполадок.)

В IPFilter по умолчанию принята стратегия разрешения. Если вы предпочитаете стратегию запрета по умолчанию, включите в ядро следующий параметр:

```
options         IPFILTER_DEFAULT_BLOCK
```

Этих параметров нет в ядре GENERIC, так как фильтрация пакетов нужна не всем людям, а некоторые из них действуют иным образом.

Примечание

Правила IPFilter можно добавить в стратегию запрета по умолчанию, но этот параметр плюс очистка[1] всех правил заблокируют удаленный доступ к системе.

IPFW

IPFW – это инструмент фильтрации пакетов, первоначально созданный BSDi. Это обстоятельство может открыть IPFW более широкий выход на свободный рынок, нежели IPFilter, и, возможно, сделать более приемлемым в ми-

[1] Под очисткой понимается операция отмены всех правил, которая обычно вызывается первой при настройке любого фильтра пакетов (firewall'а). После нее вызываются операции разрешения или запрещения определенных видов соединений. – *Примеч. науч. ред.*

ре BSD. С помощью IPFW можно выполнять замечательные операции. Например, различные сети могут казаться друг другу локальными (перемыкание Ethernet). Возможно также управление входящим и исходящим трафиком (перекрывание трафика), а также простое управление доступом, которое предоставляет любой фильтр пакетов. В состав IPFW входит сложная система контроля состояния пакетов, идущая нога в ногу с любыми программами с открытым или закрытым исходным кодом. Впрочем, система IPFW не настолько солидна, как IPFilter, поэтому здесь фильтрация пакетов будет рассмотрена на примере IPFilter.

К счастью, IPFW и IPFilter можно использовать вместе, совмещая замечательные функции перемыкания и перекрывания, реализованные в IPFW, и расширенный контроль пакетов, предоставляемый IPFilter (однако для такого совмещения потребуются дополнительные усилия). Функция перекрывания трафика, реализованная в IPFW, рассмотрена в главе 13.

Примечание

Можно остановиться только на IPFW, однако контроль состояния пакетов, реализованный в IPFW, не так хорош, как в IPFilter. Понимая конфигурирование IPFilter, можно без труда настроить IPFW. Примеры использования IPFW можно найти в */etc/rc.firewall*.

Фильтрация пакетов: принимать по умолчанию или отвергать

Ранее обсуждавшиеся стратегии принятия и запрета по умолчанию чрезвычайно важны и в фильтрации пакетов. Если выбрано принятие по умолчанию и необходимо защитить систему или сеть, то для блокирования всех возможных атак потребуется множество правил. При выборе запрета по умолчанию необходимо явно открыть «лазейки» для каждого предлагаемого сервиса. Выбрав ту или иную стратегию, необходимо скомпилировать ядро с его поддержкой.

При запрете по умолчанию очень легко полностью заблокировать удаленный доступ к системе. В конце концов, если очистить все правила фильтра пакетов, то правило, разрешающее доступ к машине, будет удалено! Я закрываю доступ к самому себе по крайней мере раз в пару лет – обычно когда не утруждаю себя размышлениями при устранении неполадок, связанных с фильтрацией пакетов. Единственное решение – проклиная себя, лезть в машину, ехать на удаленную точку, долго извиняться перед людьми, которым я причинил неудобство, и на ходу устранять проблему.

Тем не менее в большинстве случаев стратегия запрета по умолчанию будет верной. Начинающий администратор может хорошо изучить фильтрацию пакетов лишь при наличии удобного доступа к системной консоли. Если нет полной уверенности в проведенной настройке, то посылать систему фильтрации пакетов через всю страну можно лишь при наличии компетентного местного администратора или удаленной консоли.

Базовые концепции фильтрации пакетов

Вспомним из главы 5, что соединение TCP может пребывать в различных состояниях. Оно может быть открытым, открываться и т. д. Существует полный процесс «тройного рукопожатия». Пытаясь открыть соединение, клиент посылает пакет SYN, запрашивая синхронизацию. В ответ сервер посылает клиенту пакет SYN-ACK, что означает следующее: «Я получил ваш запрос на соединение. Вот базовая информация для установления соединения». Наконец, клиент отвечает пакетом ACK, что означает: «Я получил и подтверждаю вашу информацию о соединении». Каждая часть процесса «тройного рукопожатия» должна быть выполнена, чтобы соединение было действительно установлено. Правила фильтрации пакетов должны разрешать каждую часть «тройного рукопожатия», а также саму передачу данных. Разрешение на получение входящих запросов соединений бесполезно, если правила фильтрации пакетов не позволяют отправить обратно уведомление.

В 1990-х годах фильтры пакетов проверяли каждый пакет по отдельности. Если пакет отвечал правилу, он проходил дальше. Система не анализировала предыдущие пакеты и не могла определить, был ли тот или иной пакет частью легитимной транзакции или нет. Например, если пакет SYN-ACK, привязанный к адресу «изнутри» фильтра пакетов, прибывал к нему «снаружи», фильтр пакетов решал, что этот пакет должен быть ответом на пакет, одобренный ранее. Для завершения «тройного рукопожатия» такой пакет *необходимо* было принять. В результате злоумышлениники могли подделывать пакеты SYN-ACK и использовать их для обхода казавшихся надежными устройств. Поскольку фильтру пакетов не был известен отправитель пакета SYN, он не мог отклонить такие пакеты SYN-ACK на том основании, что они ложны. Когда пакеты, отправленные злоумышленниками, проникали в сеть, они инициировали ответ от того или иного устройства и выведывали информацию о системе.

Контроль состояния, введенный в большинстве современных пакетных фильтров, призван противодействовать таким атакам. Фильтры пакетов, применяющие контроль состояния, ведут таблицу, в которой отражается каждое соединение, установленное с системой. Если входящий пакет представляется частью действующего соединения, но не соответствует ему, он отклоняется. (Хотя это усложняет работу ядра, написать правила фильтрации пакетов для stateful inspection на самом деле легче.) Например, если пакет SYN-ACK прибывает к хосту с stateful packet inspection, но этот хост не посылал SYN хосту, от которого пришел SYN-ACK, то этот пакет удаляется. Фильтр пакетов должен отслеживать множество других возможных состояний, то есть это труднее, чем может показаться.

Те, кто подумал: «А-а, фильтрация пакетов – это все равно что брандмауэр», по сути правы. Слово *брандмауэр* применимо к множеству устройств, предназначенных для защиты сети. Некоторые из них довольно замысловаты. Они тщательно анализируют каждый пакет и действуют от имени каждого сервиса, который разрешают. Другие устройства можно разложить по кирпичикам. Сейчас слово брандмауэр превратилось в модное словечко из лекси-

кона маркетоидов, не имеющее точного значения. Оно подобно слову «автомобиль». Имеете ли вы в виду Gremlin 1972 года или Maserati 2002-го? Оба автомобиля нашли свою нишу, но один из них, очевидно, спроектирован в расчете на эффективность. Хотя Gremlin'ы среди брандмауэров находят свою нишу, лучше приобрести Maserati, если важна эффективность.

Итак, система FreeBSD может стать таким крепким брандмауэром, каким ее желает сделать администратор. Фильтрация пакетов – это только начало, в каталогах */usr/ports/net* и */usr/ports/security* можно найти множество приложений-представителей (proxy), которые позволят вашей системе FreeBSD стать вровень с Gauntlet или Checkpoint[1] и даже победить в состязании, затратив на десятки тысяч долларов меньше.

Реализация IPFilter

IPFilter – это фильтр, основанный на системе правил. Пакеты сверяются с последовательностью правил IPFilters по алгоритму «наилучшего соответствия» (best fit). Это означает, что пакет сопоставляется со всеми правилами (если иное специально не оговорено), начиная с общих правил и заканчивая более точными. Например, первое правило может заблокировать все, что можно, а последующие узкие правила разрешают желаемый трафик.

Общая форма такова:

```
action direction options protocol source destination options
(действие направление параметры протокол источник пункт_назначения параметры)
```

Каждое правило следует этому базовому формату, хотя не все элементы обязательны. Правила могут быть такими простыми, как `block in from any to any`. (Результат не доставит радости, поскольку будет заблокировано все, что можно, однако это правило работает.)

Рассмотрим простое правило.

```
❶ block ❷ in ❸ log ❹ quick ❺ on ed0 from ❻ any to ❻ any ❼ with short
```

Действие в этом случае – `block` (блокировать) (❶); пакеты, подпадающие под это правило, будут остановлены.

`in` – это направление движения пакетов (❷). Допустимый выбор слов – между `in` (внутрь) и `out` (наружу) (точка наблюдения находится внутри компьютера). Пакет, сопоставляемый с `in`, поступает в систему из сети.

Ключевые слова `log` и `quick` (❸, ❹) – это параметры. Пакет регистрируется в протоколе (`log`). Если пакет подпадает под правило с отметкой `quick` (немедленно), процесс сопоставления останавливается. Вспомните, что IPFilter сопоставляет каждый пакет с каждым правилом, если не оговорено иное. Ключевое слово `quick` и есть то «иное». Если пакет подпадает под правило с ключевым словом `quick`, то дальнейшее сопоставление не проводится, а выполняется действие, предписанное этим правилом.

---

[1] Есть такие коммерческие фильтры. – *Примеч. перев.*

`on ed0` обозначает интерфейс (❺). (Все системы, подключенные к сети, имеют по крайней мере два интерфейса: обратную петлю и хотя бы один сетевой интерфейс.)

Источник (`source`) и назначение (`destination`) – `any to any` (❻). Это означает, что неважно, откуда пришел пакет и куда направляется.

Пока что это правило вполне ясно: немедленно блокировать все на этом интерфейсе. Если ed0 – единственный сетевой интерфейс в системе, то не будет ни входящего, ни исходящего трафика. (Да, по крайней мере это безопасно.)

Однако последний параметр делает это правило полезным. Выражение `with short` (❼) – это ключевое слово IPFilter, которое означает следующее: «пакеты слишком малы, чтобы считаться настоящими». Безусловно, крошечные пакеты возможны, однако они чрезвычайно редки. Такой пакет наверняка является частью атаки. (Пакеты нормального размера не подпадают под это правило; они сравниваются с последующими правилами, на основании которых принимаются или отклоняются.)

Ключевые слова и конфигурации

Изучив базовый формат правил, рассмотрим ключевые слова и конфигурации, которые пригодятся для защиты сервера Интернета. Как уже было показано, общая форма такова:

```
action direction options protocol source destination packet-options
```

Каждое из этих полей описано ниже.

Action (Действие)

Два возможных действия – `block` (блокировать) и `pass` (пропустить). Блокированным пакетам проход запрещен; пропущенные пакеты сверяются со следующим правилом из списка. (Пакет может быть блокирован одним правилом, но пропущен другим, более точным.)

Direction (Направление)

Есть два допустимых значения: `in` и `out`. Входящие пакеты (`in`) поступают в компьютер из сети. Исходящие пакеты (`out`) покидают компьютер.

Options (Параметры)

Ключевое слово `log` предписывает IPFilter пропускать пакеты к программе протоколирования, ipl. (Общую информацию об ipl можно найти в разделе «Протоколирование IPFilter», а подробную – в главе 19.) Ключевое слово `quick` предписывает IPFilter остановить сопоставление и применить это правило немедленно, если пакет ему соответствует. Ключевое слово `on` говорит системе, что это правило применимо к конкретному интерфейсу, например `on fxp0`.

Protocol (Протокол)

Это любой протокол уровня IP, например `tcp`, `udp`, `skip` и т. д. Протоколы можно указывать по имени, заданному в */etc/protocols*, или по номеру протокола. Протоколу должно предшествовать ключевое слово `proto`:

```
block in proto udp from any to any
```

Source (Источник) и **Destination** (Пункт назначения)

Ключевые слова `from` и `to` показывают направление анализируемого трафика. Для защиты сервера необходимо блокировать входящий трафик. Например, если сервер имеет IP-адрес 192.168.1.8, а в системе принята стратегия запрета по умолчанию, можно задать следующее правило (обязательно используйте IP-адреса, а не имена хостов):

```
block in from any to 192.168.1.8
```

Packet-Options (Параметры пакетов)

Последние параметры описывают специальные типы пакетов. Ключевое слово `with short` описывает IP-пакеты, которые слишком малы, чтобы быть приемлемыми. Обычно они блокируются. Аналогично, ключевое слово `with ipopts` соответствует пакетам с IP Options. В большинстве случаев им незачем попадать на сервер, поэтому такие пакеты, вероятно, надо блокировать. Параметр `keep state` предписывает IPFilter автоматически пропускать пакеты, которые выглядят продолжением соединения, отраженного в таблице состояний. Эта часть позволяет создавать простые правила, охватывающие каждый этап передачи данных (чуть дальше будет представлен пример).

Примечание

Это отнюдь не полный список всех возможных опций IPFilter. При наличии интереса к тонкой настройке IPFilter обратитесь к файлу */usr/src/contrib/ipfilter/BNF*, содержащему полное описание конфигруации IPFilter.

Наконец, пакеты можно фильтровать на основе флагов. Помните состояния соединения TCP, рассмотренные в главе 5? Начальные пакеты имеют флаги SYN, ACK и SYN-ACK. С помощью этих флагов и параметра `with` можно фильтровать пакеты исходя из состояния соединения. Самые важные флаги для фильтрации: `syn` и `syn-ack`, или `S/SA`.

Задействовав эти базовые правила и параметры, можно обеспечить подходящий уровень защиты системы. В конце этого раздела будет рассмотрена их реализация и создан простой список правил для защиты типичного сервера Интернета.

Разрешение доступа к сервисам

До этого момента я рекомендовал стратегию запрета по умолчанию и показал, как заблокировать доступ к серверу. Это прекрасно, но как разрешить доступ к тем или иным сервисам?

Первый шаг – выяснить номер порта (или его каноническое имя) и протокол для каждого сетевого демона, к которому планируется открыть доступ. (Имена и номера можно найти в файле */etc/services*.)

Например, пусть в системе есть веб-сервер (HTTP), сервер POP3 и почтовый сервер. Веб-сервер запущен на порту 80, по TCP. Почтовые серверы работают через порт 25, по TCP. Наконец, POP3 использует порт 110 и TCP.

Начнем с блокирования всех подключений по TCP, которые представляют собой запросы на установление новых соединений. Для этого учтем флаги SYN и SYN-ACK:

```
❶ block ❷ in ❸ log ❹ proto tcp ❺ all ❻ flags S/SA
```

Проанализируем каждую часть первого правила. Блокируются (❶) все подключения к системе (❷). Регистрируется каждый пакет (❸). Далее правило немного сужается: разрешаются только пакеты TCP (❹), но с любого источника и к любому пункту назначения (❺). Однако правило применяется лишь к пакетам с флагами SYN или SYN-ACK (❻). Таким образом, под это правило подпадают входящие пакеты TCP с флагом SYN или SYN-ACK или запросы на установление нового соединения. Запросы на соединение будут отклонены, если последующее правило не разрешит их.

Для открытия доступа к веб-серверу служит такое правило:

```
❶ pass ❷ in ❸ quick ❹ proto tcp ❺ from any to any ❻ port = www ❼ keep state
```

Прежде всего, пропускается входящий трафик (❶, ❷). Ключевое слово `quick` (❸) означает, что пакеты, подпадающие под это правило, пропускаются немедленно (оставшиеся правила к ним не применяются). Пакет должен передаваться по протоколу TCP (❹). Он может исходить откуда угодно и направляться куда угодно (❺). Это правило подобно предыдущему, но есть и новшество: согласно */etc/services*, пакет должен приходить в порт www (❻). Когда первый пакет принят, все другие пакеты, передаваемые в рамках этого соединения, также будут приняты (❼). Пакеты, запрашивающие доступ к порту 80, будут пропущены.

Правила для разрешения доступа к pop3 и sendmail очень просты – меняется только порт:

```
pass in quick proto tcp from any to any port = pop3 keep state
pass in quick proto tcp from any to any port = smtp keep state
```

Если бы не параметр `keep state`, пришлось бы добавлять правила для учета не только начального подключения, но и ответа веб-сервера и последующего потока данных. Такие сложные правила отлаживать нелегко.

Кроме того, при написании таких правил неясно, какой порт с высоким номером выберет клиент, чтобы инициировать соединение. Правила, разработанные вручную, могут оставить «дыры» в защите и заблокировать легитимные соединения. Позволив IPFilter сравнивать каждый пакет со списком существующих соединений, можно устранить потенциальные «дыры».

Загрузка правил IPFilter

Итак, теперь ясно, как создавать необходимые правила IPFilter. Но как их загружать?

Управлять IPFilter можно с помощью ipf(8). Эта утилита может читать правила из текстового файла. (Обычно я размещаю правила в */etc/ipf.conf*.)

Ключ `-f` предписывает IPFilter читать правила из указанного файла (в данном случае из */etc/ipf.conf*):

```
# ipf -f /etc/ipf.conf
```

При изменении правил необходимо сбросить существующие правила фильтрации и лишь затем загрузить текущие. Для этого служит ключ `-F`. В противном случае пакетный фильтр будет применять все старые правила *и* все новые. Можно явно указать группу существующих правил, которую необходимо сбросить, – входящие (`i`), исходящие (`o`) или все (`a`) правила:

```
# ipf -F a
```

Чтобы минимизировать время незащищенности системы, обе операции можно выполнить одной командой:

```
# ipf -F a && ipf -f /etc/ipf.conf
```

Примечание

Таблицей правил IPFilter можно манипулировать напрямую, чтобы исключить тот крошечный период времени, когда таблица правил пуста. Однако для этого необходимо обладать большим опытом. Если материал этого раздела изучается впервые, к такому манипулированию вы пока не готовы. Освоив правила IPFilter, обратитесь к ipf(8) для получения более подробной информации.

Группирование правил

Группа правил – это набор правил, соответствующих определенному типу пакетов. Например, одна группа правил применима только к пакетам TCP, а другая – только к пакетам UDP. Некоторые правила применимы лишь к запросам на установление новых соединений, а иные – к каждому пакету. Вовсе не обязательно сопоставлять каждый пакет с каждым правилом. Группы правил IPFilter используются для проведения пакетов по оптимальному пути правил, которые оценивают лишь тот или иной тип пакетов. Таким образом, каждый пакет сравнивается с меньшим количеством правил. Значит, правила будет легче понять и отладить, а процесс сопоставления пройдет быстрее.

Правило, которое начинает группу правил, отличается оператором `head` в конце строки. В конце правил-членов этой группы стоит оператор `group`. Каждая группа правил имеет номер группы. Номера групп совершенно произвольны. Моя рекомендация: номера групп должны отстоять друг от друга по крайней мере на сотню. В таком случае между ними можно разместить подгруппы и зависимые правила. В последующих примерах будет рассмотрено, как это сделать.

Например, следующее правило отлавливает все пакеты, выходящие через интерфейс ed0, и направляет их к группе правил с номером 100.

```
pass out on ed0 all head 100
```

Пакет, соответствующий этому правилу, будет сопоставляться с каждым правилом из списка, пока не дойдет до этого правила, после чего будет све-

ряться только с правилами группы 100. Более того, пакет, не соответствующий этому правилу, не будет сверяться ни с одним правилом из группы 100. Например:

```
block out from 127.0.0.0/8 to any group 100
```

Здесь 127.0.0.0/8 – это блок адресов, зарезервированных для соединений обратной петли. Сервер не должен посылать пакеты с этих адресов по Сети. Именно такие пакеты обратной связи и блокирует это правило. Пакеты, исходящие с таких адресов, могут проходить через интерфейс обратной петли lo0.

Для блокирования всех пакетов обратной связи универсальное правило не подходит, поскольку такие пакеты необходимы для обычных операций. Однако предыдущее правило входит в группу 100. Согласно головному правилу этой группы, обработке подлежат лишь те пакеты, которые выходят через интерфейс ed0. Пакеты обратной связи никогда не выходят через сетевой интерфейс, а значит, это правило здесь подходит. Таким образом, оно защищает другие хосты от ущерба, который может причинить неверная конфигурация вашей сети, но не защитит от пакетов IP, которые кто-либо может послать якобы через интерфейс обратной петли. В конце концов, это правило проверяет только выходящие пакеты.

Как же все это совместить? Набор правил из следующего примера предназначен для сервера с одним сетевым интерфейсом xl0 и одним IP-адресом 192.168.1.200. Этот набор правил разрешает входящие соединения POP3, SMTP и Web и любые исходящие соединения. (Другие примеры правил IPFilter для различных протоколов можно найти в */usr/src/contrib/ipfilter/rules*.)

```
#блокировать «мусор», который никогда не понадобится.
❶ block in log quick from any to any with ipopts
❷ block in log quick proto tcp from any to any with short
```

Эти первоначальные правила не входят в ту или иную группу. Они – самые важные и потому применяются ко всем пакетам. Первое правило предписывает системе отклонять все пакеты, использующие IP Options (❶). Если вы не знаете, что такое IP Options, то принимать такие пакеты не надо. Стандартные серверные программы их не требуют. Второе правило – отклонять все ультракороткие пакеты TCP (❷). Эти правила присутствуют во всех конфигурациях и настоятельно рекомендуются для любых брандмауэров или фильтров пакетов.

Следующие два правила таковы:

```
#системный интерфейс обратной петли
#у себя можно делать все, что угодно
pass in quick on lo0 all
pass out quick on lo0 all
```

Согласно этим правилам, любой входящий и исходящий трафик может проходить через интерфейс обратной петли. Оптимизировать эти правила с помощью групп не требуется, поскольку это короткие правила, выполняемые немедленно (quick).

После предшествующих общих правил следует первая группа правил:

```
#эти правила управляют исходящим трафиком
❶ pass out on xl0 all head 100
❷ block out from 127.0.0.0/8 to any group 100
❸ block out from any to 127.0.0.0/8 group 100
❹ block out from any to 192.168.1.200/32 group 100
```

Эти правила управляют трафиком, который сервер посылает по сети. Первое правило очень простое: пропускать весь трафик, выходящий через интерфейс xl0 (❶). Наибольший интерес представляет последняя часть правила, где дается старт группе 100. (Номер 100 выбран произвольно.) Пакеты, соответствующие этому правилу, будут сравниваться только с правилами из группы 100.

Второе правило относится к тому же типу, что и рассмотренное ранее. Оно блокирует трафик из интерфейса обратной петли (❷). В этом правиле не надо указывать интерфейс, поскольку уже ясно, что эта группа правил применима только к пакетам, выходящим через интерфейс xl0. Говоря коротко, здесь блокируются исходящие пакеты, которые никогда не надо посылать.

Следующее правило блокирует недопустимые пакеты подобного типа (❸). Если адреса 127.0.0.0/8 являются адресами обратной петли, не следует обращаться к ним по сети.

Наконец, последнее правило группы блокирует исходящие пакеты, адресованные на наш собственный адрес (❶). Это другая сторона вопроса: пакеты, привязанные к нашей системе, не должны покидать сетевую карту.

Здесь заканчиваются правила группы 100. Если пакет выходит через этот интерфейс и не подпадает ни под одно правило группы 100, путь ему будет разрешен. Исходящие соединения этой машины устанавливаются согласно стратегии разрешения по умолчанию.

Теперь рассмотрим пакеты, поступающие в систему:

```
#эти правила управляют входящим трафиком
❶ block in on xl0 all head 200
❷ block in from 127.0.0.0/8 to any group 200
❸ block in from 192.168.1.200/32 to any group 200
❹ pass in quick proto tcp from any to any port = www keep state group 200
pass in quick proto tcp from any to any port = pop3 keep state group 200
pass in quick proto tcp from any to any port = smtp keep state group 200
#помочь клиентам закрыть соединения, когда они запрашивают сервис,
#который мы не предоставляем.
#Сервер будет казаться более быстрым, а общая нагрузка, вызываемая Интернетом,
#снизится совсем немного.
❺ block return-rst in log proto tcp from any to any flags S/SA group 200
❻ block return-icmp(net-unr) in proto udp all group 200
```

Первое правило предыдущей группы блокирует весь трафик, приходящий из Интернета через этот интерфейс (для входящего трафика используется стратегия запрета по умолчанию), и дает начало группе правил 200 – другой

произвольно выбранный номер (❶). Входящие пакеты, прошедшие через предыдущие правила, попадают именно на это правило.

Второе правило блокирует все пакеты, приходящие через интерфейс обратной петли (❷). Поскольку это правило входит в группу 200, оно применимо только к пакетам, прибывающим через сетевую карту. Есть много возможных объяснений появления таких пакетов, но все они не выдерживают критики.

Аналогично, следующее правило блокирует любой пакет, приходящий якобы с собственного IP-адреса сервера (❸). Такие пакеты никогда не должны приходить по сети. Практически единственное возможное объяснение их присутствия – попытка атаки на сервер.

Следующие три правила открывают «дыры» для сервисов, которые предлагает этот сервер (❹). Ключевое слово `quick` отсекает дальнейшую обработку, а параметр `keep state` поддерживает соединение на все время обслуживания запроса.

Последние правила (❺, ❻), возможно, наиболее трудные для понимания. Входящие сетевые запросы сохраняются до истечения тайм-аута. Первое из трех правил посылает уведомление reset- отклоняемым соединениям TCP. В нем говорится, что тот или иной запрос на соединение был отклонен, а не прекратил существование по истечении тайм-аута клиента. Последнее правило делает нечто подобное для соединений UDP (❻). Для точного описания работы этих правил потребовалось бы глубокое объяснение IP. Эти правила не только обеспечивают безопасность, но и создают обстановку, более благоприятную для других пользователей, снижая сетевой трафик. Я настоятельно рекомендую использовать их для фильтрации входящего трафика.

Типичный входящий пакет, запрашивающий веб-страницу, проходит через девять правил. Первые два правила проверяют, не слишком ли мал пакет и не содержит ли он IP Options. Очевидно, что они нефункциональны. Следующие два правила проверяют, не поступил ли пакет через интерфейс обратной петли. Далее выясняется, не отправляется ли пакет по сети. Пакет – входящий, а не исходящий, поэтому все правила группы 100 пропускаются. Наконец, пакет сверяется с правилом для пакетов, входящих через сетевой интерфейс, и попадает в группу правил 200. Здесь он сопоставляется с двумя правилами для заведомо подложных пакетов и, наконец, принимается. Не будь групп правил, пакету пришлось бы пройти через 12 правил. Может показаться, что 25% – это не так много, однако они могут стать очень важными при обработке миллионов пакетов. Чем более сложными становятся правила, тем выше экономия.

Протоколирование IPFilter

Утилита ipmon(8) – это отдельная програма IPFilter, управляющая протоколированием. IPFilter читает пакеты по мере их прохождения через ядро и передает их системе протоколирования FreeBSD как исходящие от источника LOCAL0. (В главе 19 объясняется, как правильно обрабатывать эти дан-

ные.) Самый простой способ использования ipmon состоит в том, чтобы выполнять дамп протокола для syslogd(8) (см. главу 19) с ключом -s:

```
#ipmon -s
```

Клетка

Один из самых старых механизмов защиты UNIX построен на идее измененного корневого каталога (changed root), или chroot, согласно которой пользователь ограничен подразделом файловой системы, а остальная часть файловой системы защищена. chroot подходит для маленьких сервисов, но не так полезен, если один сервер обеспечивает хостинг для десятков (или сотен!) клиентов. У каждого клиента свои потребности, а каждая программа имеет свои требования для работы с chroot.

Клиенты, которые понимают мощь UNIX, нередко предъявляют запросы, усложняющие жизнь администратора. Например, хотят установить то или иное программное обеспечение или переконфигурировать веб-сервер, чтобы включить последний крутой Apache. В двух словах, они хотят получить доступ с правами root, а в большинстве систем UNIX нельзя раздавать доступ root клиентам на сервере с множеством пользователей.

Нельзя, если у вас не FreeBSD. Администраторы FreeBSD столкнулись с этой проблемой довольно давно и решили ее, значительно улучшив процесс chroot. По сути, они решили ее так хорошо, что в системе FreeBSD можно создавать на диске целые виртуальные машины и изолировать их от остальной части системы. Такая виртуальная машина называется *клеткой* (*jail*).

Клетку можно представить как среду клиент-сервер. Основной сервер – это хост, а каждая система-клетка – это клиент. Изменения, проводимые в хосте, можно отразить на всех системах, но изменения в клетке не затронут основную систему, если только не позволить клетке заполнить весь диск или что-либо подобное.

Находясь в клетке, клиенты могут иметь доступ root и даже устанавливать любые программы по своему желанию. Это не окажет влияния на основную систему. Все процессы, запускаемые в клетке, ограничены ее средой. Ядро не предоставляет им доступ к информации, лежащей за пределами клетки. Файловая система в клетке ничего не знает о файлах и файловых системах вне клетки. Поскольку программы и процессы в клетке ничего не знают о «внешнем мире» и не могут читать или получить к чему-либо доступ за его пределами, пользователь заблокирован. Мало того, что клиент не может вырваться из клетки, – если в клетку проник злоумышленник, дальше нее он не пройдет. Такой механизм позволяет обезопасить систему и удовлетворить потребности клиентов.

На современном оборудовании, с недорогими дисками и избытком памяти, система FreeBSD может обслуживать десятки клеток с веб-серверами (однако для этого конфигурация ядра должна разрешать такое количество веб-серверов; см. главу 4). С точки зрения сбыта машина-клетка представляет

собой хорошую промежуточную стадию между виртуальным доменом на общем сервере и собственным локальным сервером.

Конфигурирование сервера для использования клеток

Перед использованием клеток необходимо убедиться, что сервер настроен правильно. Клетки налагают специальные требования на сервер. Самое досадное из них – демоны нельзя привязать ко всем доступным IP-адресам.

Каждая клетка привязана к определенному IP-адресу и определяется этим IP-адресом. Клетка должна иметь исключительный доступ к этому адресу; никто иной не может его использовать. Если в основном сервере есть демон, привязанный ко всем доступным IP-адресам системы, то этот демон воспрепятствует запуску клетки. Взглянув на вывод команды sockstat(1), можно заметить несколько записей, в которых локальный адрес походит на «*.22». Это означает, что демон «прослушивает» этот порт на всех IP-адресах. Для использования клеток необходимо заново сконфигурировать эти демоны – каждый из них должен прослушивать только один IP-адрес.

Перед запуском клетки проверьте следующие демоны.

portmap

Демон portmap – самый проблематичный из стандартных демонов FreeBSD, препятствующий комбинированию NFS и клеток. (Поскольку лишь очень немногие системы в открытом Интернете используют NFS, это обычно не вызывает трудностей.)

syslogd

syslogd – это другая история: системный регистратор (system logger) открывает сокет, а значит, может посылать зарегистрированные сообщения другим серверам. Чтобы заставить замолчать syslogd, установите параметр `syslogd_flags="ss"` в */etc/rc.conf*, однако в этом случае невозможным будет удаленное ведение протоколов. Программа syslogd подробно рассмотрена в главе 15.

named и sendmail

Другим демонам, таким как named и sendmail, необходимо подсоединяться ко всем возможным адресам. В качестве решения можно предложить привязывать их только к одному IP-адресу и запускать их на базисной системе (host system). Однако, поскольку клетки уже используются, почему бы не установить «клетку сервисов», содержащую эти демоны? Это не только проще, но и безопаснее: такая схема подразумевает дополнительный уровень защиты. (Хотя сегодня named и sendmail достаточно надежны, у них подмоченная репутация. Многие опытные администраторы чувствуют себя значительно лучше, когда эти сервисы находятся в клетке.)

inetd

inetd также подсоединяется ко всем возможным адресам, но этим аспектом его работы достаточно просто управлять с помощью ключа `-a`. Если хост с клетками имеет IP-адрес 192.168.1.222, добавьте `inetd_flags="-a 192.168.1.222"` в */etc/rc.conf*.

sshd

sshd – последний проблематичный сетевой сервис. Снова предположим, что сервер с клетками имеет IP-адрес 192.168.1.222. Указать демону sshd, какой порт «прослушивать», можно с помощью следующей записи в */etc/ssh/sshd_config*:

```
ListenAddress 192.168.1.222
```

Поскольку сам хост, на котором расположены клетки, вероятно, не предоставляет никаких сетевых сервисов, полезно выключить все сетевые демоны, за исключением sshd. В идеале вывод команды sockstat должен выглядеть примерно так:

```
# sockstat -4
USER     COMMAND    PID   FD PROTO  LOCAL ADDRESS         FOREIGN ADDRESS
root     sshd       248    3 tcp4   192.168.1.222:22      *:*
#
```

Есть только один демон, «прослушивающий» сеть, – демон sshd. Он «прослушивает» определенный порт на определенном IP-адресе (192.168.1.222). Этот демон не пересекается с клетками.

Конфигурирование ядра для поддержки клеток

В предыдущем разделе освещалась сетевая часть конфигурирования клеток, однако необходимо провести дополнительное конфигурирование ядра. Система клеток имеет три специальных sysctls:

jail.set_hostname_allowed

По умолчанию в клетке пользователь root может устанавливать имя хоста для этой клетки. Поскольку клетка использует это имя для взаимодействия с «основным» хостом, изменение имени хоста может легко запутать администратора, ответственного за управление клетками. Если установить этот sysctl в 0, изменение имени хоста будет запрещено.

jail.socket_unixiproute_only

По умолчанию с клеткой можно взаимодействовать только через IP. Хотя маловероятно, что пользователь будет применять, скажем, сокеты UNIX или IPX, это вполне возможно. Однако система клеток поддерживает только IP. Позволить пользователям применять другие протоколы – значит, позволить им «выбираться» из клетки. Скорее всего, они не воспользуются таким доступом в полной мере, однако неразумно считать себя умнее любого злоумышленника. Присвойте этому параметру 1. Этот осмот-

рительный шаг поможет крепче удерживать пользователей. Значение 0 позволит использовать любой сетевой протокол и сокеты.

jail.sysvipc_allowed

System V IPC – это стандарт UNIX для разрешения[1] межпроцессного взаимодействия (Interprocess Communication, IPC) через сегменты разделяемой памяти. По существу, связанные друг с другом программы могут использовать один участок памяти для хранения разделяемой информации. По умолчанию IPC не может применяться в клетке, так как система клеток не создает отдельные области памяти для каждой клетки. Разрешение IPC позволило бы информации просачиваться из клетки и обратно. Впрочем, воспользоваться таким слабым местом способен лишь квалифицированный взломщик. Для разрешения System V IPC присвойте этому sysctl значение 1. Многие программы баз данных требуют System V IPC.

Установка клиента

Установка клетки бесхитростна, хотя для этого потребуется дерево исходного кода FreeBSD (см. главу 6). Например, пусть необходимо создать клетку в разделе */jail1*. (Клетки могут располагаться и в каталогах, однако размещение клеток в отдельных разделах позволит управлять их размером «на скорую руку». Некоторые администраторы предпочитают наблюдать за пользователями и начислять «очки» за избыточное потребление дискового пространства.) Сначала перейдем к дереву исходного кода FreeBSD (обычно в */usr/src*). Для первой клетки запустим такую команду:

```
# make world DESTDIR=/jail1
```

Эта команда создаст полную копию FreeBSD и установит ее в каталоге */jail1*.

Для последующих клеток не надо заново собирать все исполняемые файлы; можно установить файлы, собранные в первый раз. Для этого запустим такую команду:

```
# make installword DESTDIR=/jail1
```

В результате полный набор программ userland системы FreeBSD будет помещен в клетку.

Примечание

У многих администраторов есть свои методы уменьшения пространства, занимаемого клеткой. Однако предыдущий метод апробирован. При наличии интереса к другим методам поищите их описание в архивах почтовой рассылки FreeBSD-security.

Дерево /etc

В каждой клетке есть свое дерево */etc*. Хотя не все здесь функционально, проще игнорировать лишнее, чем вычищать его. Копию дерева */etc*, взятую

[1] В оригинале allowing, но IPC – это просто стандарт межпроцессной коммуникации. – *Примеч. науч. ред.*

из исходного кода, который применялся для создания клетки, необходимо правильно установить в каталоге клетки. Именно это делают следующие команды:

```
# cd /usr/src/etc
# make distribution DESTDIR=/jail1 NO_MAKEDEV_RUN=yes
```

После каталога */etc* надо создать файлы устройств. (Поскольку для клетки не нужны все файлы устройств, необходимые для полной системы, в MAKEDEV есть специальный флажок, предназначенный для клеток.)

```
# cd /jail1/dev
# sh MAKEDEV jail
```

Многим программам нужен файл */kernel*. Даже если они ничего с ним не делают, его наличие делает их счастливее. (Доступ к реальному ядру системы предоставлять не стоит, поэтому привяжите эту подделку к безвредному объекту. Таким образом, злоумышленники сколько угодно могут переписывать ядро клетки безо всякой для себя пользы.)

```
# cd jail1
# ln -sf dev/null kernel
```

IP-адрес

После установки дерева каталогов клетке необходимо назначить IP-адрес, поскольку у каждой клетки он свой. Предположим, что 192.168.1.223 – это IP-адрес клетки. Применим ifconfig для присоединения этого адреса к сетевой карте.

```
# ifconfig fxp0 alias 192.168.1.223
```

Такое присоединение можно автоматически выполнять при начальной загрузке, если добавить в */etc/rc.conf* следующую запись:

```
ifconfig_fxp0_alias0="192.168.1.223"
```

Файловая система процессов

Наконец, каждой системе FreeBSD нужна своя файловая система процессов, или procfs. Если клетки не используются, то о procfs беспокоиться не надо, она появляется автоматически при начальной загрузке системы; ее можно настраивать, а программы легко обращаются к ней при необходимости. Однако надо построить ее инфраструктуру. Я создаю сценарий */usr/local/etc/rc.d/jail.sh* и добавляю в него все строки для монтирования procfs:

```
# mount -t procfs proc /jail1/proc
```

Теперь клетка готова.

Ввод клетки

Выполнив предыдущие шаги, воспользуйтесь jail(8) для запуска клетки:

```
# jail <path to jail> <jail hostname> <jail IP> <command>
```

Например, для базового конфигурирования тестовой клетки введите:

```
# jail /jail1 jailhost 192.168.1.223 /bin/tcsh
```

Появится приглашение командной оболочки. В этот момент вы находитесь в однопользовательском режиме в той клетке, которая запущена и работает. Оболочка, устанавливаемая по умолчанию, может быть любой. Для работы в интерактивном режиме я предпочитаю tcsh. Однако между текущим состоянием клетки и однопользовательским режимом FreeBSD есть различия. Хотя стартовая последовательность команд в клетке не выполнена, сетевые службы клетки сконфигурированы.

Некоторые команды в клетке недоступны. Например, попробуйте добавить псевдоним (alias) к сетевому интерфейсу. В ответ будет выдано сообщение об ошибке «permission denied» (нет разрешения).

Осмотритесь вокруг и попробуйте выбраться из клетки. Попробуйте перейти в каталог, который существует в системе, но находится вне каталога клетки. Вы – root; попробуйте получить доступ к процессам, которые, как вы знаете, работают в системе. Когда вы устанете биться головой о стену, исследуйте систему клетки. Мощные инструменты UNIX, такие как perl(1) и cc(1), полностью доступны. Можно даже воспользоваться cvsup и попытаться переустановить систему, хотя это не лучшая идея. (Вспомните, что ядро и userland должны быть абсолютно синхронизированы; программы userland, имеющиеся в клетке, не приведут к аварии ядра, но и не будут работать, как ожидалось!)

Процессы

Процессы в клетке не могут видеть остаток системы. На хосте работают не только клетки, но и масса других служб. Вот моментальный снимок, выдаваемый командой top в клетке, которая запущена в однопользовательском режиме. Виден процесс командной оболочки, процесс top и ничего больше. Процессы «основной» системы не видны.

```
last pid: 10578;  load averages:  0.00,  0.00,  0.00    up 1+09:21:29  19:16:49
2 processes:   1 running, 1 sleeping
CPU states:  0.0% user,  0.0% nice,  0.4% system,  0.0% interrupt, 99.6% idle
Mem: 6708K Active, 27M Inact, 23M Wired, 36K Cache, 61M Buf, 444M Free
Swap: 1024M Total, 1024M Free
  PID USERNAME PRI NICE   SIZE    RES STATE  C   TIME   WCPU    CPU COMMAND
10574 root      20    0  1432K  1116K pause  0   0:00  0.00%  0.00% tcsh
10578 root      96    0  1956K  1136K CPU1   1   0:00  0.00%  0.00% top
```

Итак, клеточная камера готова, необходимо ввести ее в эксплуатацию и немного украсить.

Заключительная настройка клетки

Процесс установки клетки не так сложен, как установка FreeBSD, выполняемая инсталлятором. Для подготовки среды клетки необходимо войти в нее и выполнить следующие команды:

1. Сначала создайте */etc/rc.conf*. Включите в него следующие строки:

   ❶ `portmap_enable="NO"`
   ❷ `network_interfaces=""`
   ❸ `sshd_enable="YES"`

   Поскольку portmap не работает в клетке должным образом, он отключен (❶). Так как система клетки будет жаловаться, если не сможет сконфигурировать интерфейс, предпишем ей игнорировать интерфейсы (❷). И поскольку после запуска клетки возникнут трудности с доступом к ней из командной строки, легче всего разрешить в клетке sshd и обращаться к ней по сети (❸).

2. Для FreeBSD нужен файл */etc/fstab*. Так как в клетке нет управления файловой системой, пустого файла будет достаточно.

   ```
   # touch /etc/fstab
   ```

3. Так как sendmail(8) будет жаловаться на отстутствие базы данных с псевдонимами, воспользуемся newaliases(1) и создадим надлежащую базу данных. (Этот шаг не нужен, если sendmail в клетке не будет запущен, или будет запущен postfix (см. главу 12), или почтовый сервер в клетке просто не требуется.)

   ```
   # newaliases
   ```

4. Зададим пароль root для среды клетки. Пароль должен отличаться от пароля хоста, – в конце концов, для этого и нужна клетка.

   ```
   # passwd
   ```

5. Пользователи будут признательны за верный часовой пояс в клетке. (По крайней мере, они смогут отсчитывать секунды в своей темнице.)

   ```
   # tzsetup
   ```

Запуск клетки

Начиная с этого момента клетка похожа на систему FreeBSD по умолчанию. В ней можно конфигурировать службу имен, добавлять пакеты, пользователей и т. д. Однако если выйти из оболочки (shell), виртуальная машина остановится, а работа клетки будет прекращена.

Примечание

Для добавления программ к серверам с клетками я настоятельно рекомендую применять пакеты (packages); сборка «портов» может отнять много времени процессора.

Впрочем, клетка может работать и в многопользовательском режиме. Для долговременной работы клетки во всем великолепии многопользовательского режима достаточно выполнить в самой клетке сценарий виртуальной машины */etc/rc* – либо вручную, либо автоматически при загрузке, добавив эту команду к сценарию */usr/local/etc/rc.d/jail.sh*.

Для запуска клетки из «основной» системы ввведите такую команду:

```
# jail /jail1 jailhost 192.168.1.223 /bin/sh /etc/rc
```

Примечание

При запуске клетки будут выданы несколько сообщений об ошибках. Большинство из них связаны с sysctls, которые не могут быть доступны в среде клетки.

Теперь клетка работает. Можно войти в нее с помощью ssh и конфигурировать клетку точно так же, как и любую другую систему.

Управление клетками

В клетках происходит сложное управление процессами. Если зарегистрироваться на сервере с клетками, можно увидеть все процессы во всех клетках. Какие из них принадлежат самому серверу, а какие – клеткам?

Команда `ps -ax`, выполненная в «основной» системе, покажет все работающие процессы, даже процессы клеток. Символ «J» в поле `STAT` означает, что процесс запущен в клетке. Если на сервере есть несколько клеток, то исходя из предназначения каждой из них можно догадаться, к какой клетке относится тот или иной процесс. Например, если есть только один сервер имен и он находится в клетке, можно смело держать пари на принадлежность процесса named.

Может возникнуть желание управлять процессами клетки, находясь за ее пределами. Однако проще всего управлять клеткой изнутри. Для этого войдите в клетку как root и применяйте `ps -ax` и все другие стандартные инструменты управления процессами, предназначенные для контроля работающих программ. Если вы не хотите входить в клетку, потребуется прибегнуть к более сложным механизмам управления.

procfs

Чтобы выяснить, в каких клетках запущены те или иные процессы, обратитесь к файловой системе процессов, procfs. (Возможно, это единственный случай, когда придется изучать */proc*, – обычно procfs используется только программами, такими как ps и top.) Такая процедура лучше всего подходит для идентификации клетки по ID процесса. Если процесс базы данных выходит из-под контроля и поглощает все больше и больше памяти, можно по PID в */proc* определить, в какой клетке запущен этот процесс, и предпринять надлежащие действия.

В */proc* есть каталог для каждого процесса, работающего в системе. (Если вам скучно, можете поискать информацию в других файлах.) Чтобы определить, к какой клетке принадлежит процесс, найдите каталог, соответствующий ID процесса, и обратите внимание на файл *status*. Последнее слово в этом файле – имя хоста клетки, в которой работает этот процесс. Если процесс запущен не в клетке, последним будет дефис (-).

Останов клетки

При останове хоста также будут остановлены различные клиентские клетки. Останов клетки без останова хоста ненамного сложнее.

Программы shutdown(8) и reboot(8) бесполезны для останова клетки, поскольку их основная обязанность состоит в синхронизации и размонтировании дисков, отсоединении от сети и т. д. В виртуальной машине это не требуется.

Для останова клетки сначала войдите в нее как root. Если в клетке обслуживаются программы, которым необходим безопасный останов (например, базы данных), для каждой из них надо запустить специальный сценарий останова.

```
# /bin/sh /usr/local/etc/rc.d/programname.sh stop
# /bin/sh /etc/rc.shutdown
```

После этого пошлите основному процессу ((1) клетки сигнал останова (сигнал 15).

```
# kill -15 -1
```

Эта программа остановит все процессы клетки. Поскольку клетка – это всего лишь процессы, в этот момент она будет остановлена.

Примечание

Не выполняйте `kill -15 -1` вне клеток, иначе будет остановлено много сервисов, а система перейдет в бесполезный однопользовательский режим.

Мониторинг системы защиты

Итак, вы считаете, что сервер в безопасности. Может быть.

К сожалению, существуют взломщики, которые не находят лучшего занятия, кроме как держать себя в курсе последних «прорех» в защите. Опираясь на эти знания, они пытаются проникнуть в систему, которая, по их мнению, уязвима. Даже если вы добросовестно читаете сообщения рассылки FreeBSD-security и накладываете все появляющиеся «заплатки», однажды защита системы может быть взломана. Хотя сказать точно, что система взломана, не удастся, следующие приемы помогут узнать о таких попытках:

- Стремитесь понимать работу своих серверов. Регулярно запускайте на них `ps -auwx`. Изучите, какие процессы обычно бывают запущены. Если тот или иной процесс не распознается, его надо исследовать.
- Обращайте внимание на открытые сетевые порты с помощью `netstat -na` и `sockstat`. Какие порты TCP и UDP должен «прослушивать» сервер? Если порт не распознается, исследуйте его. Наверное, это что-то безобидное, но это может быть и черным ходом злоумышленника.
- Необъяснимые неполадки в системе также являются подсказкой. Многие злоумышленники неуклюжи, и у них мало навыков системного администратора. Они затевают бессмысленные атаки и считают себя «крутыми».

(По-настоящему квалифицированные взломщики не только заметают следы, но и стараются не причинить вреда системе, чтобы не насторожить администратора.) Перезагрузки, происходящие без видимой причины, могут быть сигналом установки нового ядра. Также они могут свидетельствовать о неполадках в аппаратном обеспечении и неверной конфигурации, поэтому в любом случае такие факты необходимо исследовать.

Есть два инструмента защиты, которые я особенно рекомендую для понимания работы системы. Первый инструмент – это lsof(8) (*/usr/ports/sysutils/lsof*), который перечисляет все файлы, открытые на компьютере. Чтение его вывода – это обучение само по себе; возможно, вы и не догадывались, что на сервере запущено столько лишнего. Странные открытые файлы свидетельствуют либо о недостаточном понимани системы, либо о чьих-то нежелательных действиях.

Второй инструмент – nessus(8) (*/usr/ports/security/nessus*). Это автоматизированный сканер слабых мест в системе защиты. Аудит безопасности – это замечательный способ узнать, что взломщик может увидеть в вашей системе.

Если система взломана

Что делать, если система взломана? Простого ответа нет. На эту тему написаны толстенные книги. Однако вот несколько общих рекомендаций.

Прежде всего, взломанной системе нельзя доверять. Если злоумышленник получил доступ root к серверу Интернета, он может заменить любую программу в системе. Даже если закрыть «дыру», через которую он проник, в системе может остаться установленная им программа регистрации, которая будет посылать имя регистрирующегося пользователя и пароль в какой-нибудь канал IRC. Не доверяйте такой системе. Ее обновление делу не поможет, поскольку даже sysinstall и компилятор находятся под подозрением.

Не стесняйтесь написать по адресу *FreeBSD-security@FreeBSD.org* и попросить совет. Опишите ситуацию и объясните, почему вы думаете, что система взломана. Однако будьте готовы и к такому ответу: переустановите операционную систему с безопасных носителей (FTP или CD-ROM) и восстановите данные с резервной копии. (Вы ведь прочли главу 3?)

Хорошая система защиты увеличит шансы избежать взлома. Удачи.

9

Каталог /etc

Каталог */etc* содержит основные сведения о конфигурации, необходимые для загрузки UNIX. Сталкиваясь с незнакомой системой UNIX, я прежде всего присматриваюсь к */etc*.

Кратчайшая дорога от младшего «админа» UNIX к админу-середнячку – чтение */etc*. Да, надо читать все его содержимое. Да, читать придется много. Однако понимание */etc* равнозначно пониманию взаимосвязи всех компонентов системы. Так или иначе, со временем администратор UNIX постепенно осваивает эту информацию. Если добавить ее к багажу знаний в самом начале пути, дорога будет легче.

Многие файлы */etc* обсуждаются в тех главах, к которым они имеют самое непосредственное отношение (например, */etc/services* рассматривается в главе 5). Здесь же речь пойдет о важных файлах, для рассмотрения которых нет более подходящего места.

Разнообразие файлов /etc

Различные системы UNIX используют разные файлы */etc*. В большинстве случаев эти файлы есть не что иное, как переименованные или перестроенные файлы из BSD4.3 или BSD4.4. Например, первый раз столкнувшись с IBM AIX, я стал искать файл */etc/fstab*, подобный эквивалентному файлу BSD. Его там не было. Однако недолгие поиски навели меня на */etc/filesystems*, который оказался переделанной версией */etc/fstab*, учитывающей специфику IBM. Если знать о том, что эта информация содержится в */etc*, и понимать, что тех или иных файлов там, очевидно, нет, то поиски не отнимут много времени.

Даже совершенно различные системы FreeBSD имеют почти идентичные каталоги */etc*. Хотя некоторые дополнительные программы добавляют сюда свои файлы, можно ожидать, что определенные файлы будут присутствовать в любой системе FreeBSD.

Примечание

Перед изменением любого файла из каталога */etc* освежите в памяти информацию о системе управления версиями (Revision Control System, RCS), почерпнутую в главе 3. Я настоятельно рекомендую создать каталог */etc/RCS* и добросовестно использовать его для экспериментов. Изменения в */etc* могут вывести систему из строя. И хотя восстановление перепутанной таблицы файловых систем помогает превратить сведущего администратора в отличного, такой путь – один из наименее приятных для достижения этой цели.

Файлы по умолчанию

Каждый файл в каталоге */etc/defaults/* содержит присваивания переменных. Эти файлы не предназначены для редактирования администратором; их удел – подмена файлами с теми же именами из самого каталога */etc*.

Например, процесс обновления полностью заменит файлы в */etc/defaults*. Хотя каждая новая версия FreeBSD имеет слегка иную конфигурацию по умолчанию, разработчики прикладывают немалые усилия к тому, чтобы изменения этих файлов были обратно совместимы. Это означает, что вы не должны проходить по обновленной конфигурации и вручную объединять все изменения. Надо лишь проверить новый файл по умолчанию на предмет появления новых возможностей конфигурирования.

/etc/defaults/rc.conf

Один из файлов, используемых в большинстве случаев, – */etc/defaults/rc.conf*. Он содержит десятки строк такого вида:

```
"named_enable="NO"
```

Для изменения этого параметра надо редактировать */etc/rc.conf*, а не */etc/defaults/rc.conf*. Внося изменения в */etc/rc.conf*, перечислите все переменные и их требуемые значения. Запись в файле */etc/rc.conf* подменит запись в */etc/defaults/rc.conf*. (Не копируйте файл */etc/defaults/rc.conf* в */etc*! Это может стать причиной массы неполадок.)

Примечание

Хотя в процессе установки системы создается файл */etc/rc.conf*, вам вполне может понадобиться создать в */etc* другие файлы подмены.

Поняв разнообразные файлы по умолчанию, вы сможете легко оценить незнакомую систему FreeBSD. Для этого надо просто изучить соответствующие файлы подмены в */etc*.

/etc/adduser.conf

В некоторых системах UNIX создание нового пользователя – дело нелегкое. Для этого необходимо вручную отредактировать */etc/passwd*, заново пересобрать базу данных паролей, отредактировать */etc/group*, создать домашний

каталог, установить новые файлы .* и т. д. Во FreeBSD программа adduser(8) намного упрощает процедуру добавления пользователей, запуская все эти программы. В файле *adduser.conf* хранятся настройки adduser по умолчанию. Эти переменные легко установить – достаточно ввести имя, знак равенства и значение. Комментариям должен предшествовать знак «#». Вот простая запись из этого файла с соответствующим комментарием:

```
# verbose = [0-2]
verbose = 1
```

verbose

Первая запись, `verbose`, управляет объемом информации, который можно увидеть при запуске adduser. Если `verbose = 0`, adduser пригласит ввести информацию о новом пользователе, и это все, что появится на экране. Если `verbose = 1`, то adduser позволит переделать */etc/adduser.conf* перед добавлением нового пользователя. Если `verbose = 2`, adduser выдаст множество предупреждений, вопросов и другую информацию. (По умолчанию эта переменная равна 1, но ей можно легко присвоить 0, – как только процесс добавления пользователей будет освоен. В этом случае можно настраивать *adduser.conf* по своему усмотрению.)

defaultpasswd

Переменная `defaultpasswd` может быть либо yes (да), либо no (нет). Она определяет, могут ли пароли пользователей принимать значения по умолчанию. Если вы имеете учетную запись без пароля, то любой желающий, который знает имя пользователя, может подсоединиться к системе. Если безопасность хоть немного вас волнует, присвойте этой переменной значение `yes`.

dotdir

Запись `dotdir` содержит путь к образцам пользовательских файлов .*, таким как *.cshrc*, *.login* и т. д. Этот каталог содержит приемлемые файлы по умолчанию. Если необходимо создать собственные файлы .*, лучше всего создать каталог в */usr/local/share/skel*, чтобы при обновлениях системы ваши изменения не были перезаписаны (см. главу 6).

send_message

Если указать полный путь к файлу в `send_message`, adduser будет посылать каждому новому пользователю приветственное сообщение. Если этой переменной присвоить значение `no`, сообщения посылаться не будут.

Сообщение adduser использует переменные, и можно добавить собственные переменные, отредактировав */usr/sbin/adduser*. Если вы знакомы с Perl, это не сложно; если нет, лучше всего использовать предлагаемые переменные: `$username`, `$fullname` и `$password`. (Поскольку это сообщение посылается новому пользователю, посылать пароль в какой-то мере глупо и бесполезно. На самом деле пользователь ввел свой пароль, чтобы извлечь это сообщение, – значит, пароль у него есть.) При желании можно создать свое собственное

сообщение вместо того, чтобы использовать сообщение по умолчанию. Обычно я использую примерно такой */etc/adduser.message*, подставляя в него название компании:

```
$fullname, (полное имя)

Добро пожаловать в Компанию.

Помощь можно получить по телефону 800-555-1212 или в режиме онлайн по адресу http:/
/helpdesk.companyname.com.

Применение этой учетной записи определяется нашей политикой допустимого
использования, которую можно найти по адресу http://aup.companyname.com
или в файле /usr/local/share/company/aup в этой системе.

Спасибо за ваш труд. Мы будем рады помочь вам.

Группа поддержки Компании
```

logfile

Переменная `logfile` сообщает программе путь к файлу протокола, в котором фиксируется все происходящее. Подойдет значение по умолчанию.

home

Если в системе организовано необычное разделение на разделы, каталоги пользователей могут располагаться не в обычном */home*, а в другом месте. Переменная `home` задает их местоположение.

path

Если программное обеспечение расположено в необычном месте, может потребоваться изменить запись `path`. (В некоторых системах дополнительные программы находятся в */opt*.)

shellpref

Переменная `shellpref` хранит список доступных командных процессоров (shells) в порядке предпочтения. При добавлении или удалении того или иного командного процессора этот список необходимо корректировать.

defaultshell

Как можно догадаться, переменная `defaultshell` содержит командный процессор, по умолчанию назначаемый пользователю.

defaultgroup

У записи `defaultgroup` другое предназначение. Для каждого пользователя система FreeBSD назначает уникальную группу. Например, при добавлении пользователя mvlucas предпринимается попытка создать группу mwlucas (такой подход дает большую гибкость при назначении прав доступа). Если необходимо, чтобы каждый пользователь принадлежал к определенной группе, задайте здесь ее имя. В противном случае оставьте значение `USER`.

defaultclass

Строка `defaultclass` определяет, какой класс *login.conf* (см. */etc/login.conf*) по умолчанию назначается добавляемому пользователю. Эту строку можно оставить пустой или назначить один из классов, которые были ранее сконфигурированы в */etc/login.conf*.

uid_start

Наконец, переменная `uid_start` определяет номер идентификатора пользователя (user ID, UID), с которого adduser начинает нумерацию. Значение по умолчанию равно 1000. Это число можно изменить, если необходимо выдерживать соответствие между UID на разных операционных системах; например, в различных дистрибутивах Linux нумерация начинается с разных номеров. Однако если в синхронизации UID нет необходимости, на эту переменную можно не обращать внимания.

/etc/crontab

Файл *crontab* управляет планировщиком FreeBSD (*cron*), который позволяет администратору задавать время выполнения тех или иных команд системы.

Каждый пользователь имеет отдельный файл *crontab*, который можно редактировать с помощью `crontab -e`. Файл */etc/crontab* – это системный файл. В отличие от пользовательских файлов *crontab*, файл */etc/crontab* позволяет администратору задавать пользователя, от имени которого будет запущено задание. Например, он может сказать следующее: «Запускать это задание в 10 часов вечера по вторникам от имени root, а другое задание – каждый день в 7 утра от имени пользователя nobody.» Другие пользователи могут запускать задания только от своего имени.

Примечание

Файл */etc/crontab* считается системным файлом FreeBSD. Будьте внимательны и не перезапишите этот файл при обновлении системы (см. главу 6). Один из способов упростить обновление */etc/crontab* состоит в том, чтобы добавлять свои записи в конец файла, отделяя их несколькими строками знаков «#».

Операторы задания окружения

Файл */etc/crontab* начинается с нескольких операторов, задающих значения переменных среды окружения, поскольку программе cron необходимо установить окружение оболочки для приложений, которые она запускает. Если вы знакомы с программированием в среде командной оболочки, эти операторы можно изменить в соответствии с требованиями системы. Однако вносить изменения надо осторожно: записи, расположенные в верхней части */etc/crontab*, окажут влияние на все программы, вызываемые из crontab. (Для каждой команды, запускаемой из cron, переменные окружения можно указать в командной строке.)

Вот несколько типичных переменных окружения, которые задаются в */etc/crontab* в системе FreeBSD 4.5-STABLE:

```
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/
#
```

Знаки «#» – это комментарии или пустые строки, используемые для разделения записей. Они облегчают чтение файла.

Строки файла crontab, следующие за информацией об окружении, разбиты на восемь колонок, образующих таблицу. Первые пять колонок представляют время запуска команды: минута, час, день месяца, месяц года, день недели соответственно. Звездочка (*) в любой колонке означает «любой(ая)» (every one), а номер означает «именно в это время».

После колонок времени следует имя пользователя, от имени которого будет запущено задание, а затем сама команда.

Пользовательские файлы crontab почти идентичны системному – в них нет только колонки имени пользователя.

Задание времени

Значения времени в crontab должны быть верными. Общее правило таково: минуты, часы и дни недели нумеруются с 0, а дни месяца и месяцы – с 1. Кроме того благодаря старинным разногласиям между AT&T и BSD, для обозначения дня недели «воскресенье» применяется как 7, так и 0.

Например, для того чтобы пользователь dbadmin запускал программу */usr/local/bin/db-backup.sh* на 55-й минуте после каждого часа каждый день до бесконечности, строка crontab должна выглядеть так:

```
55      *     *     *      *        dbadmin /usr/local/bin/db-backup.sh
```

Звездочки предписывают программе cron запускать это задание каждый час, каждый день месяца, каждый месяц и каждый день недели.

Для ежедневного запуска этого задания в 13:55 подойдет такая строка:

```
55      13    *     *     *        dbadmin /usr/local/bin/db-backup.sh
```

Здесь 13 представляет час в сутках, а 55 – количество минут после часа.

Многие допускают при использовании cron типичную ошибку – указывают старший компонент времени, но пропускают младший. Допустим, что задание необходимо запускать ежедневно в 8 часов утра. Предположим, что для этого введена следующая строка:

```
*     8     *     *     *        dbadmin /usr/local/bin/db-backup.sh
```

Все верно, это задание будет выполняться в 8 часов утра. Но точно так же оно будет выполняться в 8:01, 8:02, 8:03 и т. д. до 9:00. Если задание выполняется больше одной минуты, такая строка быстро поставит систему на колени.

Вот правильный способ указать 8 и только 8 часов утра:

```
0      8     *     *     *        dbadmin /usr/local/bin/db-backup.sh
```

Для того чтобы указать диапазон времени, скажем, для ежечасного запуска этой программы между 8 утра и 6 вечера, нужна приблизительно такая строка:

```
1     8-18  *     *        1-5     dbadmin /usr/local/bin/db-backup.sh
```

Указывая точные моменты времени, разделяйте их запятыми:

```
1  8,10,12,15,18   *   *    1-5     dbadmin /usr/local/bin/db-backup.sh
```

Более того, можно указать доли времени, или *шаги* (*steps*). Так, для того чтобы программа запускалась каждые пять минут, введите такую строку:

```
*/5   *     *     *     *        dbadmin /usr/local/bin/db-backup.sh
```

Шаги и диапазоны можно объединять. Например, если задание должно выполняться каждые пять минут, но спустя одну минуту после предыдущего задания, то строка может выглядеть так:

```
1-56/5  *     *     *     *      dbadmin /usr/local/bin/db-backup.sh
```

Дни выполнения задания можно определять в двух полях, указывая день месяца и день недели. Если указать и день месяца, и день недели, то задание будет запущено при выполнении любого из условий. Например, программе cron можно указать следующее: «запускать это задание cron 1-го и 15-го числа плюс каждый понедельник»:

```
55    13    *     1,15    *     1       dbadmin /usr/local/bin/db-backup.sh
```

Если для задания необходимо нестандартное окружение, определите его в командной строке, подобно запуску команды в среде оболочки. Например, если программе db-backup.sh необходима переменная окружения `LD_LIBRARY_PATH`, ее можно установить так:

```
55  * * * *  dbadmin LD_LIBRARY_PATH=/usr/local/dblib ;/usr/local/bin/db-backup.sh
```

/etc/csh.*

Файлы */etc/csh.** содержат значения по умолчанию для csh и tcsh, задаваемые в масштабе всей системы. Когда пользователь входит в систему из любой из этих оболочек, оболочка выполняет все команды, найденные в */etc/csh.login*. Подобным образом, при выходе пользователя из системы выполняется */etc/csh.logout*. В */etc/csh.cshrc* можно разместить общую информацию о конфигурации оболочки.

/etc/dhclient.conf

Многие операционные системы предлагают лишь базовые средства конфигурирования клиента DHCP, которые не предоставляют возможности тонкой настройки или подгонки; либо вы используете то, что есть, либо не используете ничего. Любая операционная система, включая все системы BSD, ко-

торая применяет клиент DHCP от Internet Software Consortiumt, предоставляет возможности его тонкой настройки.

В большинстве случаев пустой файл клиента DHCP (*/etc/dhclient.conf*) дает полную функциональность DHCP, однако он работает безошибочно не во всех ситуациях. Возможно, вы подключаетесь к серверу DHCP через всю страну, или в вашей локальной сети есть неполадки, или в сети есть несколько серверов DHCP. Настройка конфигурации DHCP поможет преодолеть эти трудности. Сведения, выдаваемые по DHCP, представляют собой информацию о сетевой конфигурации, такую как IP-адрес, маршрут по умолчанию и доступные серверы имен. Без действительного, правильного назначения свойств хоста соединение с Интернетом невозможно.[1]

Записи в *dhclient.conf* похожи на код C. Обычно они включают в себя объявление переменной и ее значение. Каждая строка заканчивается точкой с запятой.

Продление запросов на владение адресом

При запуске программа dhclient запрашивает IP-адрес, который использовался (был выделен ему) последний раз, и по умолчанию в течение десяти секунд пытается получить этот адрес. Время рестарта (reboot time) – это период времени, в ходе которого клиент пытается заново получить старый адрес. Для изменения времени ожидания применяется оператор reboot. Например, я работал в больших корпоративных сетях, где сервер DHCP был в другом штате. Увеличив период времени рестарта, я легко получал предыдущий сетевой адрес. Время рестарта можно задать в *dhclient.conf*, поставив в конце строки точку с запятой – в стандартном стиле кода C.

```
reboot 20;
```

Если в течение времени рестарта клиент не сможет получить предыдущий IP-адрес, он будет запрашивать новый.

Отклонение ответов неисправных серверов DHCP

Одна из наиболее интересных особенностей dhclient состоит в возможности отклонять ответы неисправных серверов DHCP. Например, некоторые сети позволяют чуть ли не каждому подключиться чуть ли не ко всему, подобно ресурсам в выставочных залах и компаниях-разработчиках. В таких ситуациях сервер DHCP вполне может оказаться неисправным. Неверный ответ DHCP, получаемый системой, может исходить от неработоспособного сервера.

Для выявления неисправного сервера DHCP исследуйте в */var/db/dhclient.leases* информацию о выделении адреса. В этом файле перечислены все

---

[1] Если DHCP в сети настроен неверно, то сеть вообще не будет работать. Если плохо настроен сервер DHCP, все хосты будут теряться в догадках о своем адресе и прочих параметрах TCP/IP, а если неверно настроен клиент DHCP, то не сможет работать только тот хост, где это случилось. Впрочем, неверно настроить клиент DHCP – это надо постараться! – *Примеч. науч. ред.*

адреса, которые когда-либо получала система, включая нерабочие. Выявление неисправного сервера DHCP проводится методом проб и ошибок. Возьмите IP-адреса каждого сервера DHCP и отклоняйте его по одному за раз, пока не будет найдена работоспособная конфигурация. Например, если адрес неисправного сервера – 192.168.1.84, введите:

```
reject 192.168.1.84
```

Примечание

Если в сети найден неисправный сервер DHCP, намного лучше найти его и отключить, чем накладывать «заплатку» с помощью оператора reject. Впрочем, во внешних сетях обычно на это нет полномочий.

Объявление информации о хосте

Если вы находитесь не в своей сети и хотите проявить любезность к администратору, добавьте в свой *dhclient.conf* оператор `send`. Информацию, введенную в операторе `send`, сервер DHCP сохранит в базе данных о выделенных адресах. Администратор локальной сети может воспользоваться этой информацией, чтобы найти вас, если ваша система начинает плохо себя вести и нарушает работу сети. (Возможно, вы не находите в этом ничего хорошего, однако лучше позволить легко найти себя, чем быть пойманным.)

```
send host-name "mwlucas-laptop.bigcompany.com";
```

В нормальной обстановке (а зачастую и в ненормальной) другие параметры *dhclient.conf* относительно бесполезны, например возможность отклонять ответы DHCP, не содержащие необходимой информации. По-настоящему подробные сведения о более экзотических опциях dhclient можно найти в dhclient.conf(5).

/etc/fstab

Файл */etc/fstab* описывает файловые системы, присутствующие в системе. Для получения подробной информацию о таблице файловых систем обратитесь к главе 16.

/etc/ftp.*

Файлы */etc/ftp.** управляют работой сервера FTP. Подробные сведения о */etc/ftpusers*, */etc/ftpchroot*, */etc/ftpwelcome*, */etc/ftpmotd* и основных операциях FTP можно найти в описании FTP, представленном в главе 15.

/etc/hosts.allow

Файл */etc/hosts.allow* управляет доступом к демонам, скомпилированным с поддержкой TCP Wrappers. Этот файл исключительно подробно освещен в главе 8.

/etc/hosts.equiv

Файл */etc/hosts.equiv* позволяет доверенным удаленным системам подключаться и выполнять команды в локальной системе без предоставления пароля и даже без регистрации. Предполагается, что доверенные хосты выполняют аутентификацию пользователей самостоятельно, а значит, локальная система не беспокоится о повторной аутентификации.

Этот файл удобен и полезен в «дружественных» сетях, но, к сожалению, таких сетей сегодня нет. На самом деле с помощью этого сервиса любой рассерженный служащий может нарушить работу корпоративной сети. Машина с */etc/hosts.equiv*, открыто подключенная к Интернету, становится легкой добычей для первого малыша со скриптами, который на нее набредет. На самом деле */etc/hosts.equiv* и относящиеся к нему сервисы заставали врасплох даже превосходных экспертов по безопасности.

Как бы то ни было, для применения этой опасной функции необходимо включить в */etc/inetd.conf* программу rsh, или rlogin, или обе (см. главу 13). Формат */etc/hosts.equiv* прост: имя хоста, за которым следуют необязательные имена пользователей.

Предположим, есть две машины UNIX, «daffy» и «bugs».[1] Если в файле */etc/hosts.equiv* на машине bugs есть «daffy», то пользователь на daffy может получить доступ к командной оболочке без набора пароля.

```
daffy; rlogin bugs
Last login: Tue Apr  3 19:12:08 from 192.168.1.200
Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991, 1993, 1994
        The Regents of the University of California.  All rights reserved.

FreeBSD 5.0-CURRENT (PETULANCE) #0: Mon Aug 21 12:27:59 EDT 2000
You have mail.
bugs;logout
rlogin: connection closed.
daffy;
```

Видите? Без пароля. Все хорошо, если только какой-нибудь злоумышленник не проник в daffy. Работая с этим инструментом, помните, что компрометация одной машины означает компрометацию каждой машины сети. Для современного сетевого окружения программа rlogin и подобные инструменты действительно не годятся.

Сравнительно недавние версии rlogin и rsh позволяют запрашивать пароль для доступа к другой системе. Впрочем, если вы собираетесь так поступать, можно все сделать как следует и начать применять ssh (см. главу 13).

/etc/hosts.lpd

Файл */etc/hosts.lpd* – это один из самых простых файлов в */etc*. Хосты, перечисленные здесь (каждый в отдельной строке), могут отправлять задания на

---

[1] Daffy (англ.) – взбалмошный, безумный; bugs (англ.) – ошибки. — *Примеч. перев.*

печать на принтеры, которыми управляет эта машина. Имена хостов задавать можно, но неполадки DNS способны «перекрыть кислород» заданиям на печать, поэтому вместо имен следует указывать IP-адреса.

В отличие от многих других конфигурационных файлов UNIX, этот не допускает номера сетей и сетевые маски, здесь необходимо указывать отдельные имена хостов или IP-адреса.

/etc/inetd.conf

Демон inetd обслуживает входящие сетевые подключения для меньших демонов, которые не запускаются часто. Более подробную информацию можно найти в разделе «Демон inetd» в главе 13.

/etc/locate.rc

Программа locate(1) находит все файлы с заданным именем. Например, для отыскания файла *locate.rc* введите следующее:

```
# locate locate.rc
/etc/locate.rc
/usr/share/examples/etc/locate.rc
/usr/src/usr.bin/locate/locate/locate.rc
#
```

Можно увидеть, что *locate.rc* есть в трех местах: в основном каталоге */etc*, в каталоге с примерами системных файлов и в исходном коде системы.

Раз в неделю система FreeBSD сканирует свои диски, создает список всех найденных файлов и сохраняет его в базе данных. Программа, создающая такой список, использует значения по умолчанию, указанные в */etc/locate.rc* (который не оказывает влияние на выполнение самой программы locate(1)). Изменяя эти параметры, можно внести изменения в процесс создания базы данных locate, а также в ее содержимое. При этом следует руководствоваться следующим:

- Программа нахождения файлов хранит свои временные файлы в *TMPDIR*. Если в системном временном каталоге недостаточно места, этот путь можно изменить.
- Местоположение недельной базы можно изменить с помощью переменной FCODES. Однако это может оказать влияние на другие компоненты системы locate, поэтому будьте готовы к странным результатам.
- В переменной SEARCHPATHS перечислен каждый каталог, в котором требуется проводить поиск. По умолчанию это каталог /, то есть весь диск. Для того чтобы индексировать только часть диска, укажите здесь конкретное значение.
- В переменной PRUNEPATHS перечислены каталоги, которые индексировать не надо. Значение по умолчанию исключает временные каталоги, которые обычно содержат только недолговечные файлы.

- Переменная `FILESYSTEMS` задает вид файловой системы, которую необходимо индексировать. Значение по умолчанию – UFS, стандартная файловая система FreeBSD; однако можно указать и другие типы файловых систем, например MD (memory disks, диски памяти) или NFS (network filesystem, сетевая файловая система). Если смонтированы внешние файловые системы, такие как раздел EXT2FS, можно включить и их. (Между прочим, индексирование сетевых файловых систем – плохая идея; если все серверы начнут индексировать файловый сервер, сеть будет сильно перегружена.)

/etc/login.access

В некоторых серверах есть сотни пользователей, и у каждого из них свои потребности. Как назначить различные полномочия каждому из них?

FreeBSD предоставляет несколько способов управления доступом пользователей. В файле */etc/login.access* указываются пользователи, которым разрешена регистрация в системе. Каждый раз, когда вы пытаетесь установить соединение с системой FreeBSD, сначала проверяются полномочия в *login.access*. Если там есть правила, запрещающие вход в систему того или иного пользователя, то его попытка зарегистрироваться немедленно провалится. По умолчанию этот файл пуст – нет никаких ограничений для пользователей, имеющих имя и пароль.

В файле */etc/login.access* есть три поля, разделенных двоеточиями. Первое поле предоставляет (+) или отнимает (-) право на вход в систему; второе поле – список пользователей или групп; третье – список источников подключений.

Подобно файлу */etc/hosts.allow* для TCP Wrappers (см. главу 8), */etc/login.access* разрешает выражения «all» (все) и «all except» (все, за исключением), позволяя администратору задавать основные, но точные правила. Программа входа в систему проверяет правила, руководствуясь первым совпадением, а не лучшим. Когда система находит правило, которому соответствуют и группа, и источник подключения, соединение немедленно разрешается или отклоняется. Значит, порядок правил очень важен.

Например, чтобы разрешить регистрацию с системной консоли только членам группы wheel и пользователю root, можно попробовать следующее:

```
+:wheel root:console
```

Однако с этим правилом не все в порядке: на самом деле оно не отменяет полномочий пользователей на вход в систему. По умолчанию регистрация разрешена, а эта запись явно предоставляет такое право двум категориям пользователей, поэтому данное правило не будет препятствовать регистрации других пользователей.[1] При регистрации после него будут применяться следующие правила. Если мое имя – javerage, и я пытаюсь зарегистрироваться с системной консоли, это правило не запретит мне доступ.

Поэтому вместо оператора, подобного предыдущему, попробуйте другой, обратный:

---

[1] Помните главу 8? Это стратегия разрешения по умолчанию.

```
-:ALL EXCEPT wheel root:console
```

Это правило отклоняет подключения быстрее. Кроме того, снижается вероятность ошибки администратора. Как правило, при создании списков лучше запрещать регистрации, а не разрешать их.

Под это правило подпадает пользователь Joe Average (javerage) – в подключении ему будет немедленно отказано. Поскольку правила применяются по принципу первого совпадения, последующие правила ничего не изменят. Так можно избежать несанкционированного доступа к системе.

Источники подключения

Последнее поле в файле */etc/login.access*, источник подключения, может принимать совершенно разные значения. Здесь можно указывать несколько различных типов информации: имена хостов, адреса хостов, номера сетей, имена доменов, `LOCAL` и `ALL`.

Имена хостов

Имена хостов опираются на DNS или файл *hosts*. Если есть подозрение, что сервер имен подвергается атакам взломщиков, эту систему не стоит использовать; злоумышленники могут назначить имени хоста любой IP-адрес и обмануть вашу систему при подключении. Тем не менее можно предпринять следующее:

```
-:ALL EXCEPT wheel:fileserver.mycompany.com
```

Пользователи в группе wheel могут входить в систему с fileserver, а все остальные – нет.

```
ALL
```

`ALL` означает неизменное соответствие. Как будет показано далее, это особенно полезно в комбинации с `EXCEPT`.

Адреса хостов и сети

Адреса хостов похожи на имена хостов, но они невосприимчивы к мистификациям с DNS.

```
-:ALL EXCEPT wheel:169.254.8.3
```

Номер сети заканчивается точкой:

```
-:ALL EXCEPT wheel:169.254.8.
```

Такое правило позволит каждому члену группы wheel регистрироваться с машины, IP-адрес которой начинается с 169.254.8, а также запретит доступ всем остальным.

Например, если доступ к брандмауэру надо запретить всем, кроме тех, кто регистрируется с управляющей рабочей станции, можно поступить так:

```
-:ALL EXCEPT wheel:ALL EXCEPT 192.168.89.128 192.168.170.33
```

LOCAL

Самое сложное местоположение – LOCAL. Ему соответствует любое имя хоста без точки (обычно это хосты в локальном домене). Например, *www.AbsoluteBSD.com* предполагает, что любой хост в «AbsoluteBSD.com» соответствует LOCAL.

Такая функциональность реализована через обратный DNS (см. главу 12), который позволяет определять имя хоста по его IP-адресу. Мой лэптоп может заявить, что его имя – *pedicular.AbsoluteBSD.com*, но его IP-адрес соответствует записи обратного DNS, утверждающей, что он находится где-то в сети *home.com*. Машина в *AbsoluteBSD.com* подумает, что моему лэптопу присвоено имя хоста, принадлежащее домену *home.com*, а значит, лэптоп не является локальным. Таким образом, я не могу применять LOCAL в качестве метода проверки.

Как же объединить все это? Например, файл *login.access* может состоять из одной строки, которая позволит администраторам регистрироваться на сервере, а всем другим пользователям удаленные подключения будут запрещены:

```
-:ALL EXCEPT wheel:ALL
```

Однако это правило может значительно сузить операционную среду, например, если в компании есть группы сотрудников. Обычно это две группы – «dns» (сотрудники, которые могут редактировать файлы доменных зон) и «www» (сотрудники, которые могут конфигурировать веб-сервер). Для таких серверов может подойти следующий *login.access*:

```
-:ALL EXCEPT wheel dns www:ALL
```

В обычной группе веб-серверов файл *login.access* может выглядеть так:

```
-:ALL EXCEPT wheel:console
-:ALL EXCEPT wheel dns www:ALL
```

При наличии этой записи пользователи не смогут войти в систему, если их не добавить в группу, которой разрешен доступ.

/etc/login.conf

Те, кому не подходит управление по принципу «все или ничего», могут организовать более точное управление с помощью */etc/login.conf*. Этот файл позволяет настраивать окружение для отдельных пользователей и ограничивает предоставляемые им ресурсы.

Система *login.conf* подразумевает определение классов и закрепление каждого пользователя за тем или иным классом. У каждого класса есть свои ограничения на доступ к системным ресурсам. Изменения ограничений для класса затрагивают всех пользователей, которые к нему принадлежат. Класс пользователя задается при создании его учетной записи. Изменить класс можно запуском команды chsh username от имени root.

Определения классов

Каждое определение класса состоит из последовательности различных присваиваний. Когда пользователь входит в систему, login(1) проверяет эти переменные. На основе их значений устанавливаются ограничения на доступ пользователя к ресурсам, ведется их учет и настраивается окружение.

По умолчанию */etc/login.conf* начинается с класса «default», который применяется для учетных записей, не относящихся к другим классам. По сути, этот класс предоставляет пользователям неограниченный[1] доступ к системным ресурсам. Если класс «default» соответствует вашим потребностям, этот файл вообще не надо корректировать. (Если необходимо перекрыть доступ пользователям, читайте дальше.)

Каждая запись в определении класса начинается и заканчивается двоеточием, хотя формально каждая запись размещается в одной строке. Символ обратной косой черты – это символ продолжения (он означает, что компьютер должен игнорировать символ конца строки). За счет этого файл можно представить в удобочитаемом формате. Вот как может начинаться описание класса в стандартном *login.conf*:

```
default:\
        :passwd_format=md5:\
        :copyright=/etc/COPYRIGHT:\
        :welcome=/etc/motd:\
...
```

Этот класс называется *default*. Я показал три переменных этого класса (на самом деле их больше, но для понимания идеи этого достаточно). Например, переменной `passwd_format` присвоено значение `md5`. Такие переменные и присваивания есть в каждом классе. Если причислить пользователя к другому классу, который задает иное окружение, то и ощущения пользователя в системе станут другими.

Некоторые переменные *login.conf* не имеют значений; они изменяют свойства учетной записи своим наличием. Например, переменная `requirehome` просто должна быть в определении класса, чтобы возыметь свой эффект.

```
        :requirehome:\
```

Вносимые изменения оказывают влияние

После редактирования *login.conf* необходимо обновить базу данных login, чтобы изменения вступили в силу:

```
# cap_mkdb /etc/login.conf
```

По умолчанию файл */etc/login.conf*, поставляемый вместе с FreeBSD, включает в себя несколько классов пользователей. Для того чтобы понять, какие

---

[1] Не столь неограниченный, как пользователю root. Доступ любого пользователя определяется, прежде всего, правами доступа к файлам и процессам в системе. – *Примеч. науч. ред.*

ограничения необходимо наложить на пользователей в различных ситуациях, обратитесь к этому файлу. В следующем разделе можно узнать о том, что должно быть установлено в этом файле.

Ограничения на ресурсы

Ограничения на ресурсы позволяют управлять объемом системных ресурсов, которые одновременно может задействовать один пользователь. Если к машине подключаются несколько сотен пользователей, а один из них решает компилировать 30 Мбайт исходного кода, то этот пользователь может потреблять намного больше памяти и времени процессора, чем ему положено по справедливости. Если ограничить ресурсы, которые один пользователь может монополизировать одновременно, то система будет более отзывчивой к остальным пользователям.

Зачастую ограничения на ресурсы привязаны к каждому процессу, и при назначении лимитов это надо учитывать. Если каждый процесс получает 20 Мбайт памяти, а каждый пользователь может запускать 20 процессов, то ограничения не играют роли, так как каждый пользователь получает 400 Мбайт памяти. Каждый класс пользователя может иметь свои ограничения на ресурсы.

В табл. 9.1 описаны переменные файла *login.conf*, отвечающие за ограничения ресурсов.

Таблица 9.1. Переменные login.conf для ограничения ресурсов

Переменная	Описание
`cputime`	Максимальное время процессора, которое может использовать любой процесс
`filesize`	Максимальный размер одного файла
`datasize`	Максимальный объем памяти, который может потреблять один процесс для хранения данных
`stacksize`	Максимальный объем стека, доступный одному процессу
`coredumpsize`	Максимальный размер дампа памяти
`memoryuse`	Максимальный объем памяти, который процесс может заблокировать
`maxproc`	Максимальное количество процессов, которые могут быть одновременно запущены одним пользователем
`openfiles`	Максимальное количество открытых файлов на один процесс
`sbsize`	Максимальный размер буфера сокета, который может задействовать приложение пользователя

Текущие и максимальные ограничения на ресурсы

В системе можно задать *текущие* (*current*) и *максимальные* ограничения на ресурсы. Текущие ограничения (`-cur`) обычно носят консультативный ха-

рактер, и пользователь может подменять их по своему желанию. (Это удобно в окружении, основанном на совместной работе, когда пользователи охотно разделяют ресурсы.) Максимальные ограничения (-max) абсолютны, и пользователь не может их раздвинуть. С помощью текущих ограничений можно предупредить пользователя о том, что он пытается выйти за пределы ресурсов, выделенных ему в соответствии с принятыми нормами.

Для задания текущего ограничения добавьте -cur к названию ограничения. Для установления жесткого лимита добавьте -max. Например, для установления текущего ограничения на количество процессов, которые могут быть у одного пользователя, выполните следующее:

```
maxproc-cur: 30
maxproc-max: 60
```

Если не указаны ни -cur, ни -max, то указанное значение будет присвоено как текущему, так и максимальному лимитам.

Задание параметров среды по умолчанию

Параметры среды по умолчанию можно задать в файле */etc/login.conf*. Это лучше, чем устанавливать их в пользовательских файлах *.cshrc* или *.profile*, поскольку эти настройки окажут влияние на все учетные записи пользователей, подключающихся сразу же после задания этих параметров.

Каждое поле окружения распознает два специальных символа: тильду (~) и знак доллара ($). Тильда замещается домашним каталогом пользователя, а знак доллара – именем пользователя. Например, в классе default строка, в которой переменной окружения MAIL присваивается значение /var/mail/$, становится строкой, в которой за «/var/mail» следует имя пользователя. Подобным образом ~bin в записи пути указывает на каталог *bin* в домашнем каталоге пользователя.

В табл. 9.2 определены типичные переменные окружения.

Таблица 9.2. Типичные переменные окружения в login.conf

Переменная	Описание
hushlogin	Если присутствует в определении класса, системная информация не выдается при входе в систему
ignorenologin	Если присутствует в определении класса, пользователь может войти в систему, даже когда файл */var/run/nologin* существует
manpath	Путь по умолчанию для поиска страниц руководства
nologin	Если присутствует, пользователь не может войти в систему. Это идентично записи в */etc/login.access*, описанной ранее
path	Путь по умолчанию для программ
priority	Приоритет процессов по умолчанию, или nice (см. главу 18)
requirehome	Если присутствует в определении класса, то для входа в систему пользователь должен иметь действительный домашний каталог

Таблица 9.2 (продолжение)

Переменная	Описание
setenv	Список переменных окружения по умолчанию
shell	Полный путь к командному процессору, который будет запущен после входа в систему. Эта запись подменяет командный процессор, указанный в */etc/passwd*. Однако переменная окружения $SHELL будет указывать на командный процессор, заданный в */etc/passwd*, поэтому окружение будет противоречивым. Изменение значения этой переменной – замечательный способ раздосадовать пользователей
term	Тип терминала по умолчанию. Чуть ли не каждая программа, пытающаяся установить тип терминала, подменяет эту запись
timezone	Значение по умолчанию переменной окружения $TZ. Пользователи могут перезаписать это значение
umask	Значение umask по умолчанию (см. builtin(1)). Пользователи могут перезаписать это значение
welcome	Полный путь к файлу, содержащему приветственное сообщение для пользователей, принадлежащих этому классу. Значение по умолчанию – */etc/motd*. (Различные приветственные сообщения могут предоставлять инструкции и извещения для разных типов пользователей.)

Управление свойствами пароля и входа в систему

В */etc/login.conf* можно задавать различные свойства пароля и входа в систему. В отличие от настройки окружения, многие из этих свойств можно установить только в этом файле. Вот некоторые свойства аутентификации.

minpasswordlen

Задает минимальную длину пароля. Это значение возымеет эффект лишь при следующем изменении пароля пользователем; никакой проверки на соответствие паролей этой длине проведено не будет. Следующий образец очень разозлит пользователей:

```
:minpasswordlen=28:\
```

passwd_format

Задает алгоритм, применяемый для шифрования паролей в */etc/passwd*. Значение по умолчанию – md5. Другие допустимые значения – des и blf (blowfish). DES наиболее полезен, когда необходимо иметь одинаковые пароли на компьютерах с различными операционными системами. По сути, blowfish – это «крутой» алгоритм, однако он необязателен, если не требуется разделять файлы паролей между системами FreeBSD и OpenBSD.

mixpasswordcase

Если это свойство задано, то при следующем изменении паролей пользователи не смогут задавать в них только строчные буквы.

`copyright`

Определяет полный путь к файлу, содержащему информацию об авторских правах на эту систему.

`host.allow`

Пользователи в классе с этим значением могут применять rlogin и rsh, чтобы подключаться к этому серверу с указанных хостов. Это разрешение подобно разрешению, предоставляемому файлом */etc/hosts.allow*. (Это не делает rlogin или rsh безопасными, поэтому такая установка настоятельно не рекомендуется.) Данная запись представляет собой список значений, разделенных запятыми. Звездочку (*) можно применять как групповой символ (wildcard) для указания сетей или доменов. Система должна фигурировать как в */etc/hosts.allow*, так и в этой записи.

```
:host.allow=192.168.1.*:\
```

`host.deny`

Здесь перечислены удаленные хосты, к которым нельзя подключиться с использованием учетных записей этого класса. Эта функциональность перекрывает файл */etc/login.access*, позволяя запретить подключение с определенными учетными записями с конкретных IP-адресов.

Если установки в *host.deny* противоречат установкам в *host.allow*, то *host.deny* имеет преимущественное значение. Как и в *host.allow*, здесь можно использовать звездочку (*) как групповой символ для указания целых сетей или доменов. С любого хоста, не указанного в *host.deny*, могут подключаться пользователи этого класса.

`times.allow`

Определяет промежуток времени, когда пользователь может входить в систему. Для задания времени необходимо в полях, разделенных двоеточиями, указать дни и периоды времени. Дни обозначаются двумя первыми буквами названия дня (Su, Mo, Tu, We, Th, Fr и Sa).[1] Время указывается в стандартном 24-часовом формате. Например, если пользователю разрешено входить в систему только по средам между 8.00 и 17.00, подойдет такая запись:

```
times.allow=We8-17:\
```

`times.deny`

Определяет промежуток времени, когда пользователю нельзя входить в систему. Отметим, что эта запись не «выкинет» пользователя, если он уже вошел в систему. Формат записи такой же, как и в *times.allow*. Если *times.allow* и *times.deny* перекрываются, то *times.deny* имеет преимущественное значение.

---

[1] Воскресенье, понедельник, вторник, среда, четверг, пятница, суббота. – *Примеч. перев.*

Функции учета

В файле */etc/login.conf* можно установить различные функции учета ресурсов. Для работы этих функций система учета должна быть включена. Учет ресурсов не настолько важен сегодня по сравнению с временами, когда недорогой компьютер стоил десятки тысяч долларов, поэтому учет ресурсов в этой книге не рассматривается. Тем не менее полезно знать, что такая возможность существует.

/etc/mail/mailer.conf

FreeBSD предоставляет несколько почтовых серверов на выбор. Файл *mailer.conf* позволяет без лишней суеты указать почтовую программу, которая будет использоваться в системе.

По традиции единственным почтовым сервером, доступным во всех системах UNIX, был sendmail(8). Поэтому многие дополнительные программы предполагают наличие */usr/sbin/sendmail* и ожидают, что он будет работать определенным образом. Поскольку эти программы написаны в расчете на sendmail, то пришедшие на замену почтовые серверы обычно принимали те же опции командной строки, что и исходный sendmail. Более того, они устанавливались как */usr/sbin/sendmail*, поэтому программы продолжали работать.

Остается одна сложность с совместимостью sendmail: в незнакомой системе администратор не знает, какая программа скрывается за */usr/sbin/sendmail*! Если кто-то установил различные почтовые серверы, чтобы с ними поэкспериментировать, необходимо на время стать детективом и уповать на удачу в распознавании так называемой программы sendmail.

Файл */etc/mail/mailer.conf* помогает навести здесь порядок, игнорируя */usr/sbin/sendmail* как почтовую программу. Ну а sendmail – это всего лишь маленькая программа, которая проверяет *mailer.conf* и переадресовывает запрос указанному почтовому серверу.

Примечание

Будучи одной из программ, унаследованных из ранних систем UNIX, sendmail работает поразному в зависимости от того, под каким именем она вызывается. Наиболее употребительные варианты имен для sendmail – это sendmail, mailq и newaliases.

Файл *mailer.conf* просто содержит список названий программ, а также путь к программе, которая вызывается на самом деле. Например, почтовый сервер Postfix (описан в главе 14) устанавливается как */usr/local/sbin/sendmail*. Соответствующая запись в *mailer.conf* выглядит так:

```
sendmail        /usr/local/sbin/sendmail
send-mail       /usr/local/sbin/sendmail
mailq           /usr/local/sbin/sendmail
newaliases      /usr/local/sbin/sendmail
```

/etc/make.conf и /etc/defaults/make.conf

make означает создание программы на машинном языке из ее исходного кода. Еще этот процесс известен как *компиляция* (*compiling*). (Она обсуждается в главе 10.) Этим процессом управляют файлы *make.conf*. *make.conf* – это одна из наиболее сложных и интересных функций BSD; файл */etc/make.conf* управляет процессом сборки программного обеспечения в локальной системе.

Подобно немногим другим конфигурационным файлам FreeBSD, *make.conf* на самом деле представляет собой два файла: */etc/defaults/make.conf* и */etc/make.conf*. Как и другие файлы в */etc/default*, файл */etc/default/make.conf* не предназначен для редактирования напрямую, поэтому при обновлении этот файл можно благополучно перезаписать.

По умолчанию все строки в */etc/default/make.conf* закомментированы. Для установки чего-либо необходимо скопировать соответствующую строку из *make.conf* в */etc/make.conf* и удалить знак «#», чтобы ее раскомментировать.

В большинстве случаев параметры в */etc/defaults/make.conf* описывают методы оптимизации, которые не применяются по умолчанию в силу той или иной причины. В определенных ситуациях одни методы приводят к созданию плохого кода, а другие просто замедляют процесс сборки. Некоторые параметры не являются обязательными и не поддерживаются разработчиками FreeBSD. (Хотя приводимые образцы безопасны; при изучении параметров в */etc/defaults/make.conf* обращайте пристальное внимание на все замечания.)

Многие параметры в */etc/defaults/make.conf* должны изменять лишь администраторы, хорошо знакомые с процессом сборки, принятым во FreeBSD. Однако довольно много параметров являются безопасными. Здесь будут рассмотрены основные параметры из их числа.

Примечание

Хотя представленные образцы взяты непосредственно из системы FreeBSD 4-stable, проверьте */etc/defaults/make.conf* на предмет любых изменений. Это не та область, где будут приятны сюрпризы.

`CPUTYPE=i686`

По умолчанию компилятор собирает программы, не оптимизируя в зависимости от типа процессора, хотя такая оптимизация может значительно улучшить производительность. Хорошим примером является комплект OpenSSL, из которого FreeBSD берет криптографические функции для программы SSH и относящихся к ней программ.

В табл. 9.3 перечислены типы процессоров, которые FreeBSD распознает в 32-разрядных системах X86.

Таблица 9.3. Типы процессоров, распознаваемые в 32-разрядных системах X86

Тип процессора	Описание
k7	Процессор AMD k7
k6-2	Процессор AMD k6-2
k6	Исходный процессор AMD k6
k5	Процессор AMD k5
p3	Pentium 3
p2	Pentium 2
i686	Обычный Intel Pentium 2 или лучше
i586/mmx	Исходный Pentium с MMX
i586	Исходный Pentium
i486	Процессор класса 486
i386	Процессор класса 386

`CFLAGS= -O -pipe`

Задает параметры оптимизации для сборки программ, не относящихся к ядру. Образец, показанный в файле по умолчанию, годен к применению и поддерживается Проектом FreeBSD. Хотя некоторые администраторы могут порекомендовать другие настройки, любые другие опции Проектом FreeBSD не поддерживаются. Если вы знакомы с другими бесплатными версиями UNIX, вам могут быть известны некоторые малопонятные параметры оптимизации.

Вообще, код FreeBSD лучше всего компилируется без дополнительных параметров. Их применение лишь ухудшит производительность. Если при использовании тех или иных нестандартных флагов программа работает со сбоями, вернитесь к стандартным флагам и пересоберите программу.

`COPTFLAGS= -O -pipe`

Параметры оптимизации `COPTFLAGS` используются только для сборки ядра. Повторюсь: установки, отличные от принятых по умолчанию, могут привести к созданию неработоспособного ядра.

`INSTALL=install -C`

По умолчанию при установке собранной программы система FreeBSD записывает новый исполняемый файл поверх старого. Параметр `install -C` заставляет инсталлятор сравнивать новую версию программы со старой. Если они идентичны, то новый исполняемый файл не устанавливается. За счет этого можно ускорить процесс обновления и сократить количество операций записи на диск. Последнее обычно не имеет большого значения, но при необходимости этим можно воспользоваться.

Параметры make.conf для обновления системы

Следующие параметры могут быть полезны при проведении обновления с исходного кода (см. главу 6). Если в этом компоненте нет необходимости, то отказ (с помощью задания соответствующих параметров) от сборки этих компонентов уменьшит время, необходимое для сборки системы. Так, можно отказаться от сборки sendmail, если он не нужен в системе.

Также эти парметры полезны при замещении части системы. Например, если в системе запущена последняя версия named из коллекции «портов», вы могли заменить */usr/sbin/named* на самостоятельно исправленную версию. Во избежание ее потери при обновлении системы необходимо указать системе, чтобы она не пересобирала эту программу.

Примечание

Если заданы параметры, предписывающие не пересобирать часть системы, обновление не залатает «дыры» в обеспечении защиты в программах, не подлежащих повторной сборке. Это означает, что на диске останутся старые, небезопасные программы. Если их запустить, то в системе безопасности появятся бреши. При необходимости отказа от сборки той или иной программы лучше всего пройтись по системе и удалить соответствующие программы.

`ENABLE_SUIDPERL=true`

Suidperl – специальная версия Perl(1), которая не устанавливается по умолчанию. Эта опция необходима для установки setuid-версии Perl в ходе обновления с исходного кода.

`PPP_NOSUID=true`

Существует давнишнее соглашение о том, что setuid-программы вредны и от них следует избавиться. Однако программе ppp, используемой для подключения к Интернету, свойство setuid необходимо, чтобы разрешить многочисленным пользователям работать с ней. Если связь с Интернетом устанавливает только пользователь root, то параметру `PPP_NOSUID` можно присвоить значение `true`.

`NO_CVS=true`

CVS – это система параллельных версий (Concurrent Versions System), используемая опытными системными администраторами. Данный параметр предотвращает сборку или установку этой системы при обновлении с исходного кода.

`NO_BIND=true`

BIND – это сервер DNS по умолчанию (см. главу 12). Если в системе установлен другой сервер, задайте этот параметр.

`NO_FORTRAN=true`

Язык программирования Фортран популярен в научных кругах и не так популярен в сообществе сетевых администраторов. Установите этот параметр, если в поддержке Фортрана нет необходимости.

`NO_LPR=true`

LPR – это система печати. Если компьютер не использует принтер или установлена особая программа печати, то можно установить этот параметр. Однако будьте внимательны: в прошлом в защите LPR было несколько «дыр».

`NO_MAILWRAPPER=true`

Mailwrapper – это программа выбора почтового сервера, переадресующая вызовы sendmail соответствующим программам (см. раздел «/etc/mail/mailer.conf»). Если в системе не запущен почтовый сервер или действуют установки FreeBSD по умолчанию, то этот параметр можно установить.

`NO_MODULES=true`

Данный параметр предотвращает автоматическую сборку модулей ядра вместе с самим ядром. Не устанавливайте его, если вам не доставляет удовольствия наблюдать за аварией системы при загрузке. Но если устанавливаете, то должны точно знать, для чего вы это делаете.

`NO_OBJC=true`

Этот параметр отменяет включение поддержки Objective C. Если в применении Objective C нет необходимости, то этот параметр можно установить. Очень немногие «порты» не работают должным образом при такой настройке.

`NO_OPENSSH=true`

Установите этот параметр, если в системе есть особая пара клиент/сервер SSH. В противном случае соберите OpenSSH.

`NO_OPENSSL=true`

OpenSSL – это пакет шифрования, используемый OpenSSH и другими сервисами безопасности. Если не запущена ни одна из программ шифрования, то можно установить этот параметр. Однако настоятельно рекомендуем ее не устанавливать, поскольку она делает невозможной сборку OpenSSH.

`NO_SENDMAIL=true`

Если в системе есть особая почтовая программа или не запущен почтовый сервер, то данный параметр можно установить. В примере главы 14 sendmail будет заменен на postfix. Однако если установить эту опцию, то в системе останется старая и, возможно, небезопасная версия sendmail. На самом деле лучше собрать всю систему.

`NO_SHAREDOCS=true`

В каталоге */usr/share/docs* системы FreeBSD располагается документация из оригинального «выпуска» BSD4.4. Кроме того, в состав системы входят более свежие документы, описывающие новые возможности. Если в данной системе документация не используется, то с помощью этого параметра можно сберечь какое-то количество процессорного времени и сэкономить на обращениях к диску.

`NO_TCSH=true`

Предотвращает сборку */bin/csh* и */bin/tcsh* во время обновления исполняемых файлов. Если в системе есть сценарии csh, которые могут исказиться при обновлении, либо командные оболочки csh/tcsh вообще не используются, установите этот параметр. (Как бы то ни было, программирование в csh – идея неудачная.)

`NOCRYPT=true`

В дополнение к коду шифрации OpenSSL, упомянутому ранее, в состав FreeBSD входит код для шифрования паролей, хеш-файлов и т. д. Если в таком коде нет необходимости, установите данный параметр. Однако, как правило, это не лучшая идея, если вы не знаете, для чего это делаете.

`NO_GAMES=true`

Предотвращает сборку программ в */usr/games*. Они не изменяются часто, поэтому без них, вероятно, можно обойтись.

`NOINFO=true`

В каталоге */usr/share/info* системы FreeBSD есть множество документов, которые не изменяются часто. Установка этого параметра позволит избежать обработки этих документов при обновлении из исходного кода.

`NOLIBC_R=true`

Установка этого параметра предотвращает обновление версии libc, содержащей только повторно входимые[1] функции (при обновлении из исходных текстов). Если вы не знаете, что это значит, не устанавливайте этот параметр!

`NOPERL=true`

Определяет, будет ли собираться интерпретатор Perl и относящиеся к нему библиотеки. Установка этого параметра может значительно ускорить

---

[1] Повторно входимая, или чистая, или реентерабельная (re-entrant) функция – это функция, которая (1) не содержит внутри себя статических переменных, хранящих состояние процесса обработки данных (функция без «памяти»); (2) получает значения параметров только через свои аргументы (но не через глобальные статические переменные); (3) возвращает значения только через аргументы либо как значение функции (через `return`). Такая функция может параллельно (или псевдопараллельно) использоваться несколькими «потоками» обработки информации в программе без какого-либо непредвиденного влияния этих «потоков» обработки друг на друга. Первый пункт требований позволяет функции не зависеть ни от какого конкретного процесса обработки данных, т. к. она не «помнит» обработанных ею ранее данных и не строит свое поведение в зависимости от них. Вторые два пункта – это требование обязательного перечисления в заголовке функции всех без исключения путей передачи данных в функцию и из нее (интерфейс функции). Это лишает функцию «побочных эффектов», не предусмотренных программистом при ее вызове (программист обычно смотрит только на заголовок функции и не выискивает «тайные» связи функции с программой через глобальные переменные, если только это специально не оговорено). Подробнее см. *http://www.opennet.ru/docs/RUS/bogatyrev/gl_1_8.html*. Хрестоматия программирования на С в UNIX. – *Примеч. науч. ред.*

процесс сборки системы, но также оставит устаревшую версию Perl. Как-никак, иногда в состав обновления FreeBSD входит и новая версия Perl! Некоторые программы, например adduser(8), написаны на Perl. Эти сценарии могут изменяться (с учетом преимуществ обновленного Perl). Если не обновлять Perl, работа этих программ нарушится. Если в системе применяются сценарии Perl, зависящие от функций конкретной версии Perl, то данный параметр можно установить.

`NOPROFILE=true`

Предотвращает сборку библиотек профилирования. Повторюсь: если вы не знаете, что это такое, не устанавливайте этот параметр.

`NOSECURE=true`

Этот параметр исключит сборку всех программ в каталоге *src/crypto*, включая Kerberos, OpenSSH, OpenSSL и другие относящиеся к ним компоненты. Не устанавливайте его на рабочем сервере Интернета.

`NOSHARE=true`

Предотвращает сборку всех программ в каталоге *src/share*, включая все старые файлы, все страницы руководства, все примеры и вообще документацию. Этот параметр можно установить, если наличие новой документации в системе не имеет значения.

`NOUUCP=true`

UUCP – это UNIX-to-UNIX Copy Protocol, старый стандарт передачи данных между машинами, который применялся до широкого распространения TCP/IP. Часть UUCP используется для последовательных консолей (см. главу 20).

`MODULES_WITH_WORLD=true`

При каждой сборке ядра модули ядра собираются по умолчанию. Данный параметр предписывает системе собирать модули при обновлении их самих, а не при обновлении ядра. Это позволит сэкономить массу времени, если между обновлениями вы экспериментируете с различными ядрами. Еще это может стать причиной непонятных проблем. Применение этого параметра требует внимательности.

`NOMANCOMPRESS=true`

В ходе обновления происходит сжатие страниц документации. Когда вы обращаетесь к странице документации, система прежде всего должна ее разархивировать. Если на диске достаточно места и вы хотите немного ускорить доступ к страницам документации, данный параметр позволит устанавливать страницы документации в неархивированном формате.

`COMPATxx=yes`

Позволяет устанавливать системные библиотеки предшествующих версий FreeBSD. Надо лишь заменить `xx` в `COMPATxx` на соответствующую версию библиотеки (`COMPAT1X`, `COMPAT20`, `COMPAT21`, `COMPAT22`, `COMPAT3X` или `COMPAT4X`). Это необходимо лишь в том случае, если в системе есть исполняемые файлы, собранные для предшествующей версии FreeBSD, и они

нужны в новой системе. При желании можно использовать несколько записей COMPATXX.

Примечание

Параметр COMPATxx=yes не подразумевает сборку библиотек из исходного кода. Скорее, библиотеки в uuencode просто хранятся в дереве исходного кода. Если один раз вы провели обновление из исходного кода при «включенной» библиотеке compat, ее можно благополучно удалить.

Параметры «портов» make.conf

Следующие параметры управляют сборкой дополнительных компонентов. Работа некоторых программ радикально меняется в зависимости от значений этих параметров. Так, некоторые дополнительные программы очень велики, и можно предписать системе не устанавливать их ни при каких обстоятельствах. В другой раз можно проинформировать систему о наличии того или иного компонента.

NO_X=true

Один из самых больших «портов» во FreeBSD – X Window System. На нее опираются многие «порты». Если данному параметру присвоить значение true, то эти «порты» не будут пытаться собирать X как часть зависимостей (dependencies). А это означает, что некоторые «порты» вообще не будут собраны, но, так или иначе, они бесполезны без X.

Некоторые части базовой системы (особенно doscmd) по умолчанию включают в себя методы X Window System. Если в системе нет X и ее наличие не предполагается, а также нет намерения установить X как зависимость к любой другой программе, установите этот параметр.

NOPORTDOCS=true

Различные программы в коллекции «портов» имеют подробную документацию, которая обычно устанавливается в каталоге */usr/local/share*. Если этот параметр установлен, то документация устанавливаться не будет. Если документация есть на рабочей машине или рабочей станции, данный параметр можно установить на серверах.

HAVE_MOTIF=yes

Motif – это библиотека графических модулей, к которым обращаются многие «порты», чтобы предоставить различные графические «штуковины». В течение нескольких лет Motif был очень дорог, однако бесплатные версии доступны в */usr/ports/x11-toolkits/lesstif* и */usr/ports/x11-toolkits/open-motif*. Если в системе есть любой из этих пакетов, установите параметр HAVE_MOTIF.

/etc/master.passwd

Этот файл содержит имена пользователей и пароли. При входе пользователя в систему набранный им пароль сравнивается с паролем в файле.

Файл */etc/master.passwd* настолько важен, что для его редактирования существует специальная программа. vipw(8) вызывает текстовый редактор, указанный в переменной окружения `$EDITOR`, позволяет вносить изменения и проверяет синтаксис файла перед тем, как принять изменения. Кроме того, vipw обновляет базу данных паролей. Это позволяет избежать многих основных ошибок, однако если вы и впрямь хотите испортить */etc/master.passwd*, vipw усложнит эту задачу, но не остановит вас.

Когда vipw позволит сохранить внесенные изменения, он также заново создаст файл */etc/passwd*. Этот файл может читать только root.

Если информация в */etc/master.passwd* противоречит другим файлам, программы обычно предполагают, что */etc/master.passwd* верен. Например, */etc/group* иногда не отображает главную группу пользователя. Главная группа, представленная в */etc/passwd*, верна, – даже если ее нет в */etc/group*.

Многим программам нужен доступ к информации, хранимой в */etc/master.passwd*, например, данные о командных оболочках и домашних каталогах должны быть общедоступны. Вместо того чтобы позволять всем желающим читать этот файл и восстанавливать зашифрованные пароли, FreeBSD предоставляет общий доступ на чтение файла */etc/passwd*, в котором хранится часть информации из */etc/master.passwd*.

Поля

Каждая строка в */etc/master.passwd* содержит десять полей, разделенных двоеточиями. Они описаны ниже.

Имя пользователя

Первое поле в */etc/master.passwd* – имя пользователя. Это либо учетная запись, созданная администратором и используемая существующим пользователем, либо пользователь, созданный при установке системы для предоставления системных сервисов. В состав FreeBSD входят различные учетные записи, например root, toor, daemon, games, uucp и т. д. Каждый из этих пользователей владеет некоторой частью системы. Различные программы можно запускать от имени этих пользователей.

Зашифрованный пароль

Второе поле – это зашифрованный пароль. Системные пользователи обычно не имеют пароля, поэтому с их именами нельзя подключиться. В учетных записях пользователей здесь располагается строка из случайных на вид символов.

Один из простых способов временно заблокировать учетную запись пользователя состоит в том, чтобы отредактировать файл паролей и поставить перед паролем звездочку (*). Хотя учетная запись по-прежнему будет активна, никто не сможет ею воспользоваться для входа в систему. Я с большим эффектом проделывал это, когда заканчивались деньги на счете у какого-либо клиента; вскоре после неудачной попытки войти в систему раздавался звонок этого пользователя.

ID пользователя

Третье поле – идентификатор пользователя, или UID. У каждого пользователя есть уникальный UID.

ID группы

Аналогично четвертое поле представляет собой идентификатор группы, или GID. В главе 7 мы уже говорили, что это главная группа пользователя. Как правило, GID такой же, как и UID, а имя группы совпадает с именем пользователя.

Класс пользователя

Следующее поле – класс пользователя (определяется в */etc/login.conf*). Класс пользователя можно изменить, запустив vipw и напрямую отредактировав *master.passwd*, либо с помощью chsh(1).

Истечение срока действия пароля

Шестое поле определяет срок действия пароля. Если оставить это поле пустым или не использовать систему учета, действие паролей не будет прекращаться. Значение этого поля задается в секундах, начиная с начала эпохи (начало эпохи – 0 часов 1 января 1970 года).

Количество секунд с начала эпохи

Аналогично, седьмое поле – количество секунд с начала эпохи до времени полного истечения срока действия учетной записи. Если в системе не применяется учет ресурсов, это поле бесполезно.

Gecos

Поле *gecos* содержит настоящее имя пользователя, номер офиса, рабочий и домашний телефоны, разделенные запятыми. Не используйте в этом поле двоеточия – они специально зарезервированы для разделения полей в самом */etc/master.passwd*.

Домашний каталог пользователя

Девятое поле – домашний каталог пользователя. По умолчанию он находится в */home*, но его можно разместить где угодно. При изменении этого поля необходимо переместить фактический домашний каталог в новое место.

Командная оболочка пользователя

Наконец, десятое поле задает командную оболочку пользователя. Если это поле пусто, система предполагает, что пользователь получает старый неинтересный */bin/sh*.

/etc/motd

Файл *motd*, или *message of the day* (сообщение дня), отображается на экранах пользователей, когда они входят в систему. В этом файле можно разместить системные извещения или другую информацию, которую должны увидеть пользователи, интерактивно входящие в систему (shell users). Заметим,

что правом на просмотр этого файла управляет параметр `welcome` в */etc/login.conf*. В системе может быть несколько файлов сообщений – по одному для каждого login-класса.

/etc/mtree/*

Файлы */etc/mtree* используются при обновлении системы и не оказывают влияния на ее повседневный запуск. mtree(8) строит иерархии каталогов, обычно для того, чтобы автоматизированный инсталлятор мог размещать в них программы. В редактировании этих файлов нет необходимости, но знать, почему они здесь находятся, полезно.

/etc/namedb/*

Файлы */etc/namedb* управляют системным сервером имен. Работа этих файлов подробно описана в главе 12.

/etc/newsyslog.conf

Этот файл настраивает ротацию и удаление файлов протоколов. Системный регистратор (system logger) подробно описан в главе 19.

/etc/passwd

Многим программам необходим доступ к такой информации о пользователях, как командная оболочка, настоящее имя и т. д. В старых системах UNIX эти данные наряду с фактическим зашифрованным паролем хранились в файле */etc/passwd*, который мог читать любой желающий. Это вызвало трудности, когда система UNIX стала распространяться в университетах. Студенты, изучающие вычислительную технику, находили очень забавным вскрытие зашифрованных паролей и, к сожалению, часто преуспевали в этом. Избрали */etc/passwd* своей целью и хакеры. В конечном итоге зашифрованные пароли были перемещены в */etc/master.passwd*. Файл */etc/passwd* остался источником информации для других программ.

Файл */etc/passwd* создается из */etc/master.passwd* удалением полей класса, изменения и истечения срока действия. Зашифрованный пароль замещается звездочкой. Вот остающиеся поля:

- имя пользователя;
- пароль (звездочка);
- номер идентификатора пользователя ID number;
- номер идентификатора группы;
- имя;
- домашний каталог;
- командная оболочка.

В разделе «/etc/master.passwd» эти поля описаны подробно.

/etc/periodic.conf и /etc/defaults/periodic.conf

Файл */etc/periodic.conf* – еще один файл в */etc*, имеющий файл по умолчанию в */etc/defaults*. periodic(8) запускается каждый день для выполнения повседневных операций по обслуживанию системы. Это источник сообщений о состоянии, которые каждый день посылаются пользователю root. Кроме того, эта программа может выполнять различные задачи, представленные в виде сценариев оболочки в */etc/periodic*. (По умолчанию periodic пытается запустить довольно много задач, в которых может и не быть необходимости; впрочем, сценарии обычно разумны и накладывают минимальную нагрузку на систему.) Каждая функция, доступная программе periodic, включается или отключается в *periodic.conf*.

periodic запускает программы ежедневно, еженедельно или ежемесячно. Для каждой группы программ есть свои значения параметров, например программы, запускаемые ежедневно, конфигурируются отдельно от программ, запускаемых ежемесячно. Этими параметрами управляют записи в файле */etc/periodic.conf*. Вот несколько стандартных записей из этого файла:

```
periodic_conf_files="/etc/periodic.conf /etc/periodic.conf.local"
```

Предыдущая строка сообщает программе periodic путь к файлам конфигурации, которые надо использовать вместо стандартных. Размещать свои файлы конфигурации можно в каталоге, отличном от */etc/*. Многие системы монтируют свои файловые системы root с доступом «только для чтения», поэтому при необходимости эти файлы можно разместить в другом месте.

`daily_output="root"`

Этот параметр сообщает программе periodic, куда посылать результаты ежедневных проверок. Если задать имя пользователя, эти данные будут отправлены ему. Если нет пользователя, ответственного за чтение почты periodic, лучше всего оставить значение по умолчанию и перенаправлять почту пользователю root, чью почту вы читаете. В качестве альтернативы здесь можно разместить имя файла. Тогда periodic будет записывать результаты в этот файл – аналогично записи в файл протокола. В этом случае newsyslog (см. главу 19) может осуществлять ротацию протокола periodic.

`daily_show_success="YES"`

Если для `daily_show_success` установлено значение `yes`, в ежедневные сообщения будет включена информация обо всех успешных проверках.

`daily_show_info="YES"`

Если для `daily_show_info` установлено значение `yes`, в ежедневные сообщения будет включена общая информация о запущенных командах.

`daily_show_badconfig="NO"`

В ежедневное сообщение будет включена информация о командах periodic, выполнить которые не удалось. Обычно это безвредные команды, поэтому если параметр `daily_show_badconfig` имеет значение `no`, потеряно бу-

дет не так уж много. Впрочем, при наличии интереса ко всему происходящему этому параметру можно присвоить значение yes.

Каждый сценарий в «ежедневных», «еженедельных» и «ежемесячных» каталогах, расположенных в */etc/periodic*, имеет краткое описание в самом верху. Просмотрите эти строчки. Параметры, которые по умолчанию включены, подходят в большинстве случаев, однако в некоторых системах можно включить и дополнительную информацию. В */etc/periodic.conf* для каждого сценария есть элемент управления, с помощью которого можно разрешать или запрещать выполнение сценариев. Их список непрерывно пополняется. Все, что я здесь перечислю, может устареть еще до передачи рукописи в редакцию, не говоря уже о времени, которое пройдет до момента получения книги вами. Поэтому я не буду вдаваться в подробности работы различных сценариев.

/etc/printcap

Файл */etc/printcap* управляет принтером. Без преувеличения, для принтеров существуют десятки и десятки параметров – от расхода на страницу до команды прогона формата для подачи нового листа бумаги. (Все параметры обсуждать не будем, но основы управления принтером рассмотрим.)

Система печати UNIX делает предположения о принтере. Определив переменные в */etc/printcap*, можно сообщить системе печати о том, как аппаратное обеспечение вашей системы отличается от классических принтеров UNIX, применявшихся двадцать лет назад. Как можно догадаться, эти отличия весьма существенны. (К счастью, большинство принтеров понимают PostScript. Это значительно упрощает их эксплуатацию.)

Если в сети, к которой подключена машина FreeBSD, есть сервер печати, то, вероятно, будет использоваться именно он. (В главе 21 есть примеры конфигураций.) При тонкой настройке могут потребоваться более экзотические параметры, предоставляемые FreeBSD. Частично параметры, которые могут потребоваться в современных системах, будут рассмотрены в следующем разделе.

Работа с записями printcap

Каждый принтер имеет собственную запись в */etc/printcap*. Все эти переменные позволяют создавать очень длинные строки, и продолжение записи на следующей строке обозначается символом обратной косой черты «\». Переменные разделяются двоеточиями. В случае применения обратной косой черты для повышения удобочитаемости записей вторая и последующая строки должны иметь двоеточия в начале и в конце выражений, в которых присваиваются значения переменным.

Первая запись в */etc/printcap* – имя принтера. Если у принтера есть несколько имен, например «ThirdFloor», «AccountingOffice» и «BigLaserJet», надо перечислить все имена, разделив их символом вертикальной черты «|».

Примечание

Во FreeBSD и в большинстве других версий UNIX имя принтера по умолчанию – «lp». Различные программы предполагают наличие принтера с таким именем. Проще всего назначить это имя предпочтительному принтеру.

Вслед за именем принтера перечислите переменные, описывающие именно его. Полный список содержит массу переменных, которые никогда не понадобятся. Здесь будут рассмотрены переменные, которые либо широко применяются, либо могут быть полезны в современных сетях. Для начала обратимся к простому примеру записи printcap и посмотрим, как ее определить. Затем будут рассмотрены переменные, позволяющие проводить тонкую настройку работы принтера.

```
lp|SalesPS|ThirdFloorPrinter:\
        :rp=SalesPS:\
        :rm=printserver:\
        :sd=/var/spool/output/lpd:\
        :lf=/var/log/lpd-errs:
```

У этого принтера есть имя lp, а также SalesPS и ThirdFloorPrinter. Удаленное имя принтера, по которому к нему обращается сервер печати, – SalesPS. Сервер печати – это машина печати, у которой есть имя в сети TCP/IP. Задания на печать хранятся в */var/spool/output/lpd*, пока они находятся в процессе обработки. Ошибки печати протоколируются в */var/log/lpd-errs*. (В главе 21 приведены рекомендации по настройке принтера.)

Следующий параметр – один из наиболее распространенных для печати в UNIX:

```
:ct=120:
```

Это тайм-аут сетевого соединения. Параметр `ct` можно применять для управления тайм-аутами удаленных сетевых принтеров. Если принтер не отвечает, служба печати ждет `ct=x` секунд перед извещением о неудаче. Значение по умолчанию составляет 120 секунд, и это слишком много для современных локальных сетей. Обычно, если принтер не отвечает в течение 30 секунд, налицо неполадки. Если же печать осуществляется на устройстве, расположенном на другом континенте, это значение надо увеличить до 240.

```
:fo=false:
```

Параметр означает «form feed upon open» (подача страницы при открытии). Если он установлен, то перед выполнением каждого задания на печать система будет выдавать чистый лист.

```
:if=/usr/libexec/lpr/input-filter:
```

Как показывает предыдущая строка, система FreeBSD может предварительно обрабатывать запросы на печать, которые она получает по сети. Такая возможность позволяет выполнять ловкие трюки, например, можно заставить надоедливый струйный принтер работать как принтер PostScript.

В */usr/ports/print/apsfilter* приведен замечательный пример, объясняющий, как это сделать.

```
:lf=/var/log/printername:
```

Этот параметр позволяет указать файл протокола для конкретного принтера.

```
:lp=/dev/lp:
```

Эта строка определяет имя печатающего устройства. Значение по умолчанию верно для устройства с одним принтером, подключенным к системе через параллельный порт. Если это сетевой принтер, создайте для него пустую запись, например, такую `:lp=:`. При наличии нескольких параллельных принтеров следует указать имя устройства для каждого из них.

```
:mx=1000:
```

Этот параметр задает максимальный размер задания на печать в блоках по 512 байт. Если выполняется печать графики, этому параметру можно присвоить значение 0 (размер не ограничен).

```
:of=/usr/libexec/lpr/filter-program
```

Полный путь к предпечатному фильтру печати. С помощью данного параметра FreeBSD поддерживает возможность предварительной обработки результатов, выдаваемых принтеру. Это может быть полезно, скажем, для преобразования PostScript в обычный вывод – подобно тому, как это делает пакет apsfilter. Специальные фильтры вывода есть для многих языков, например, в состав FreeBSD входят фильтры вывода для русского языка, а другие фильтры будут добавляться по мере предоставления их пользователями.

```
:rm=printserver:
```

Эта запись определяет имя хоста, закрепленное за удаленным сервером печати, к которому подключен этот принтер. Предполагается, что на таком сервере запущен lpd, который готов принять задание. (Большинство серверов печати, даже Windows 2000, могут взаимодействовать с lpd.)

```
:rp=printerName:
```

Это имя, которое сервер печати использует для того или иного принтера. Оно необходимо лишь при работе с сервером печати.

```
:sd=/var/spool/lpd:
```

Это каталог для спулинга, где программа печати хранит файлы при их обработке. Для каждого принтера нужен отдельный каталог спулинга.

/etc/profile

Файл */etc/profile* содержит учетную информацию, задаваемую по умолчанию для оболочки /bin/sh, подобно тому как файлы */etc/csh.** хранят информацию для пользователей csh/tcsh. bash и другие модификации sh также используют этот файл – при каждом входе в систему пользователи sh наследу-

ют значения, задаваемые в этом файле. Однако пользователь может подменить эти значения своим файлом *.profile*.

Хотя csh – это стандартный командный процессор FreeBSD, sh и его модификации (особенно bash) довольно популярны. Стоит задуматься о синхронизации настроек в */etc/profile* и */etc/csh.login*. Примеры в */etc/profile* и в */etc/csh.login* идентичны и могут служить хорошей отправной точкой.

/etc/protocols

В главе 5 были кратко рассмотрены сетевые протоколы. Файл */etc/protocols* содержит список протоколов Интернета, которые известны системе FreeBSD. Каждому протоколу назначен номер, а различные программы используют эти номера, чтобы определить, как следует обрабатывать соединения.

Почти все соединения Интернета реализуются по IP, TCP или UDP. Большинство пользователей и не догадывается, что существуют десятки протоколов и что IP – это протокол 0, TCP – протокол 6, а UDP – протокол 17. Одни протоколы широко применяются в специфическиых окружениях, а другие настолько устарели, что вы, вероятно, никогда с ними не столкнетесь. Системному администратору не надо знать все тонкости каждого протокола, однако он должен понимать, что мир не ограничен рамками TCP/IP, и иметь общее представление о других протоколах.

В файле */etc/protocols* есть отдельная строка для каждого протокола. Первая запись в строке – это официальное имя, например tlsp в примере ниже. Второе поле – это номер протокола (в этом примере – 56). Далее расположены все псевдонимы этого протокола, такие как TLSP. Наконец, комментарии отделяются с помощью знака «#».

Вот фрагмент файла */etc/protocols*:

```
tlsp        56      TLSP            # Протокол безопасности транспортного уровня
skip        57      SKIP            # SKIP
ipv6-icmp   58      IPV6-ICMP       # ICMP для IPv6
ipv6-nonxt  59      IPV6-NONXT      #  ipv6 без следующего заголовка
```

Поднимите руку, если вы когда-нибудь слышали о любом из этих протоколов. Да, именно так я и думал. Впрочем, об этом не стоит беспокоиться; помните, вы не должны знать все на свете – достаточно знать, где найти ту или иную информацию.

Примечание

Список в */etc/protocols* поддерживается Проектом FreeBSD. Вам, вероятно, никогда не потребуется его редактировать (если только для какой-нибудь особой программы не понадобится своя запись).

/etc/pwd.db

Файл */etc/pwd.db* – это база данных, представляющая собой версию */etc/passwd*. Она существует потому, что компьютерная программа получает до-

ступ к ее данным намного быстрее, чем при разборе текстового файла. Файл */etc/pwd.db* открыт для чтения, а поскольку в */etc/passwd* специально не включается секретная информация, такой подход совершенно безопасен.

Наличие файла */etc/pwd.db* объясняет, почему при редактировании файла паролей важно применять такие инструменты, как passwd(8) и vipw(8). Каждый из них автоматически запускает pwd_mkdb(8) после собственного завершения. pwd_mkdb пересобирает базу данных паролей. Если файлы */etc/passwd* и */etc/pwd.db* не синхронизированы, в системе обнаружатся странные ошибки, связанные с пользователями.

Если только вы не хакер, взламывающий базы данных, при изменении этого файла ни при каких обстоятельствах не используйте инструменты баз данных. Пусть инструменты для поддержки файлов паролей делают свою работу.

/etc/rc

Когда бы система ни загружалась, в определенный момент, когда уже можно выполнять команды userland, она запускает сценарий */etc/rc*. Этот сценарий командной оболочки монтирует все файловые системы, «поднимает» сетевые интерфейсы, конфигурирует файлы устройств, устанавливает разделяемые библиотеки и выполняет все остальные задачи, необходимые для загрузки системы.

Таких задач очень много, и некоторые из них нужны не во всех системах. Так, если в системе не используются последовательные порты, не надо запускать сценарий для их конфигурирования. Аналогично в системе, которая не подключается к сетям ATM, не надо запускать сценарий, конфигурирующий средства их поддержки.

Вместо единого сценария, выполняющего все операции, в некоторых системах стартовые задачи распределяются между сценариями поменьше, что позволяет уменьшить их размер и упростить отладку. Когда сценарию */etc/rc* необходимо установить параметры сети, он запускает */etc/rc.network*.

Вот другие сценарии командной оболочки, используемые сценарием */etc/rc*:

`/etc/rc.atm`

Конфигурирует средства поддержки сети ATM (Asynchronous Transfer Mode, асинхронный режим передачи). (Если в машине FreeBSD есть карта ATM, то вы об этом знаете – вероятно, она стоит дороже, чем все остальные компоненты системы, вместе взятые.)

`/etc/rc.diskless1` и `/etc/rc.diskless2`

Если система FreeBSD запускается без жесткого диска, будут выполнены эти сценарии. Это интересная функция FreeBSD, однако здесь она не обсуждается, поскольку для ее реализации необходим опытный администратор UNIX. Впрочем, усвоив материал этой книги, вы будете готовы к бездисковым операциям.

`/etc/rc.firewall`

Содержит основную конфигурационную информацию для ipfw. При использовании ipfw этот сценарий можно применять в качестве образца.

`/etc/rc.firewall6`

Содержит конфигурационную информацию ipfw, необходимую для IPv6.

`/etc/rc.i386`

Любая система FreeBSD на архитектуре X86 – стандартной платформе «Intel PC» – выполняет этот сценарий при загрузке.

`/etc/rc.isdn`

Если в систему встроена поддержка тарифицируемой сети ISDN, используйте этот сценарий. Вероятно, он не нужен для поддержки нетарифицируемой сети ISDN (например, для большинства сетей ISDN в США), однако он важен в Европе, Австралии и в других зонах, где соединение ISDN оплачивается поминутно.

`/etc/rc.network`

Отвечает за конфигурирование сети.

`/etc/rc.network6`

Выполняет конфигурирование сети IPv6.

`/etc/rc.serial`

Устанавливает разумные значения по умолчанию, поэтому просто работать последовательные порты будут сразу. (Конфигурировать их можно бесконечно, а без определенной настройки последовательные порты могут работать не так, как от них ожидается.)

`/etc/rc.syscons`

Этот короткий сценарий устанавливает параметры терминала, консольную заставку экрана, раскладку клавиатуры и другие параметры консоли.

`/etc/rc.sysctl`

Устанавливает значения sysctls из */etc/sysctl.conf* (см. главу 4).

Сценарий */etc/rc* конфигурируется через */etc/rc.conf* и */etc/defaults/rc.conf*. По сути, в */etc/rc.conf* устанавливаются переменные, определяющие работу */etc/rc*. Здесь же указываются сценарии, запускаемые из */etc/rc*, и описывается порядок загрузки системы.

Примечание

Если вам интересно, как система FreeBSD конфигурирует компоненты при загрузке, обратитесь к */etc/rc*. В файле */etc/defaults/rc.conf* найдите переменную, отвечающую за ту или иную функцию системы. Поищите в сценариях */etc/rc*, где применяется эта переменная, – это нетрудно сделать с помощью команды `grep -i` *`имя_переменной`* `/etc/rc.*`. Найдя этот файл, изучите его. В соответствующем разделе rc-сценария можно найти команду, которую следует применить для настройки той или иной функции. Останется лишь обратиться к странице документации и почитать об этой команде.

/etc/rc.conf и /etc/defaults/rc.conf

Файл */etc/defaults/rc.conf* огромен. Он содержит довольно много переменных, часто называемых *knobs* (кнопки), *tunables* (элементы настройки) и даже *tunable knobs* (настраиваемые кнопки). Здесь не рассматриваются все эти переменные – не только потому, что knobs постоянно добавляются (такой список немедленно устареет), но также потому, что довольно много переменных не находят широкого применения на серверах.

В стандартной системе FreeBSD переменной rc.conf может быть почти все – от раскладки клавиатуры до поведения TCP/IP. При наличии трудностей с knobs проверьте */etc/defaults/rc.conf* или rc.conf(5) – не изменилось ли что-нибудь.

В следующих разделах будут рассмотрены типичные записи из */etc/rc.conf*. Каждая запись представлена в */etc/defaults/rc.conf* один раз. Редактирование той или иной записи означает ее замену в */etc/rc.conf*.

Здесь представлены только самые распространенные параметры. Файл */etc/defaults/rc.conf* содержит, без преувеличения, сотни возможных параметров. Страница руководства rc.conf(5) будет полезна при проведении тонкой настройки системы.

Параметры запуска

Следующие несколько параметров *rc.conf* управляют конфигурированием самой системы FreeBSD и тем, как она запускает другие программы.

После того как выполнены все стартовые задачи, скрипт */etc/rc* проверяет каталоги, перечисленные в следующей переменной. Исполняемые файлы, найденные в этих каталогах (обычно это сценарии командной оболочки), будут выполнены. Вот эта переменная:

```
local_startup="/usr/local/etc/rc.d /usr/X11R6/etc/rc.d"
```

Большинство «портов», запускаемых при начальной загрузке, устанавливают свои стартовые сценарии в */usr/local/etc/rc.d*. Если программы устанавливаются в другом месте, измените этот путь, указав соответствующий стартовый каталог.

Файлы, перечисленные в следующей переменной, являются «дополнительными» файлами *rc.conf*:

```
rc_conf_files="/etc/rc.conf /etc/rc.conf.local"
```

Можно добавить дополнительные файлы *rc.conf*. Например, один из файлов *rc.conf* можно разделять между несколькими системами. В нем могут быть определены сетевые сервисы и поведение системы. В другом *rc.conf* может задаваться IP-адрес и имя хоста для конкретной системы. Такое распределение может значительно упростить общее администрирование большой группы серверов.

В то время как файл */etc/rc.conf* поддерживается централизованно, файл */etc/rc.conf.local* предназначен для локальной системы. Можно даже создать

собственные файлы *rc.conf* и разместить их в произвольном месте. Не исключено, что это не лучшая идея, но в необычной ситуации появившиеся в результате дополнительные возможности могут пригодиться.

Сетевые параметры

Они управляют настройкой сетевых функций FreeBSD во время начальной загрузки.

`hostname=""`

Параметр `hostname` задает полное имя домена системы, например «www.absolutebsd.org». Имя домена должно устанавливаться во время начальной загрузки – оно необходимо для надлежащей работы программ.

`tcp_extensions="NO"`

С течением времени TCP менялся. В RFC 1323 некоторые изменения и дополнения к протоколу TCP были объединены в «TCP Extensions» (Расширения TCP). Некоторые приложения могут воспользоваться преимуществами расширений TCP, но многие программы на это не способны. По умолчанию параметры FreeBSD имеют консервативные, блокирующие значения. Присвойте этому параметру значение `"YES"`, если приложение требует этого.

Примечание

Если один хост использует расширения TCP, а другой – нет, производительность может снизиться. По этой причине расширения TCP не рекомендованы для веб- и FTP-серверов. Большинство современных систем используют расширения RFC 1323, но, может быть, не надо усложнять жизнь старейшим системам Сети. Отсев 10% потенциальных пользователей – далеко не лучшая идея.

`log_in_vain="NO"`

Управляет протоколированием в */var/log/messages* всех попыток подключиться к системе через порты TCP или UDP, на которых ничего не «прослушивается». Этот протокол сообщает о фактах сканирования портов и исследования сети, но также собирает много мусора. Иногда этот параметр интересно установить на какое-то время, чтобы узнать о происходящих событиях, однако если протокол не исследуется регулярно, значение `"YES"` вызовет лишь трату процессорного времени и дискового пространства. В последних версиях FreeBSD значения `"YES"` и `"NO"` заменены на `"0"` и `"1"` соответственно.

`tcp_drop_synfin="NO"`

Для использования этого параметра необходимо компилировать ядро с поддержкой `TCP_DROP_SYNFIN`. В `GENERIC` этого параметра нет.

Данный параметр определяет, надо ли отбрасывать пакеты, в которых установлены флаги `SYN` и `FIN`. Программы, сканирующие сеть, используют такие пакеты для идентификации удаленных операционных систем. Однако отбрасывание пакетов SYN+FIN нарушает спецификацию TCP и мо-

жет вызвать странные неполадки в сети. При наличии неполадок отключите этот параметр и посмотрите, не исчезнут ли они.

`icmp_drop_redirect="NO"`

Переадресация сообщений ICMP используется в локальных сетях для уведомления серверов о появлении дополнительных сетевых шлюзов. Хотя существуют допустимые варианты использования ICMP, этот протокол широко используется хакерами. Если в сети переадресация ICMP не применяется, установка этого параметра незначительно усилит защиту системы. О том, применяется ли переадресация ICMP, можно спросить у системного администратора. А если вы *и есть* системный администратор и не можете ответить на этот вопрос, то вы ее не применяете.

`icmp_log_redirect="NO"`

Если в сети применяется переадресация сообщений ICMP и необходимо проводить мониторинг таких переадресаций, установите этот параметр. Тогда они будут протоколироваться в */var/log/messages*.

Количество переадресаций ICMP, которые можно протоколировать, не ограничено. Если система подвергается атаке либо в сети есть неполадки, то жесткий диск может быстро заполниться такими сообщениями. Применяйте этот параметр с осторожностью!

`network_interfaces="auto"`

Эта переменная содержит список сетевых интерфейсов, которые показывает ifconfig. Если в системе есть необычный сетевой интерфейс, то возможно (но маловероятно), что возникнут неполадки с интерфейсами, не сконфигурированными при загрузке. В этом случае попробуйте вручную перечислить интерфейсы в этой переменной, например так: `"lo0 ep0 wi0"`. Разбираться с интерфейсами – это обязанность системы, но умение преодолевать такие трудности может оказаться полезным.

`ifconfig_lo0="inet 127.0.0.1"`

Перечислите в этом параметре все сетевые интерфейсы и сведения об их конфигурации (каждый интерфейс описывается отдельной строкой). Замените `lo0` на правильное название интерфейса. Например, чтобы при загрузке присвоить сетевой карте ep0 IP-адрес 192.168.1.200 и сетевую маску 255.255.255.0, введите такую строку:

```
ifconfig_ep0="inet 192.168.1.200 netmask 255.255.255.0"
```

Если в сети применяется DHCP, присвойте интерфейсу значение `dhcp`.

`ifconfig_ep0_alias0="inet 192.168.1.201"`

FreeBSD позволяет назначать сетевой карте сотни IP-адресов. Один IP-адрес является главным, а другие – псевдонимами. Перечислите каждый псевдоним в такой форме:

```
ifconfig_interfacename_aliasnumber="inet IP address"
```

Номера псевдонимов должны быть последовательными и начинаться с 0. Если в нумерации есть разрыв, псевдонимы, следующие за ним, не будут

установлены в ходе начальной загрузки (Это обычная проблема; при ее наличии проверьте список псевдонимов.)

Параметры коммутируемого соединения PPP

В файле *rc.conf* есть несколько параметров для обслуживания коммутируемых соединений PPP. В данной книге PPP не обсуждается подробно, но об этих параметрах надо знать, чтобы воспользоваться ими при необходимости.

`ppp_enable="NO"`

Если этому параметру присвоено значение `"YES"`, система будет автоматически запускать программу ppp. (Тем не менее эта программа по-прежнему требует конфигурирования.)

`ppp_mode="auto"`

Для этого параметра есть четыре варианта:

- `"auto"` предписывает системе автоматически подключаться к Интернету при поступлении запроса.
- `"dedicated"` применяется для систем с выделенным соединением. Не подходит для выделенных телефонных линий, а предназначено для последовательных линий к другим компьютерам.
- `"direct"` предназначен для систем, на которые дозваниваются через модем. (Подробности см. в ppp(8).)
- `"ddial"` предназначен для доступа по выделенным телефонным линиям. Применяется в системах, подключенных к Интернету по коммутируемой линии. Если на стороне телефонной компании происходит обрыв соединения, система автоматически произведет повторное подключение.

`ppp_nat="YES"`

Програма ppp имеет встроенный сервис преобразования сетевых адресов (Network Address Translation). Установите этот параметр, если машина FreeBSD используется как сетевой шлюз, а соединение коммутируемое.

`ppp_profile="papchap"`

В файле */etc/ppp/ppp.conf* можно определить различные профили коммутируемых соединений. `"papchap"` – это профиль по умолчанию.

Разнообразные сетевые демоны

В состав FreeBSD входит ряд небольших демонов, обслуживающих соответствующие сервисы. Конфигурировать их можно с помощью *rc.conf*.

`syslogd_enable="YES"`

Если необходимо вести системный протокол, установите этот параметр. Программа syslogd подробно рассмотрена в главе 19.

`syslogd_flags="-s"`

Значение `syslogd_flags`, задаваемое по умолчанию, означает, что к syslogd могут подключаться только серверы, указанные в параметре `syslogd_flags`

(разрешенные). По умолчанию это предотвращает подключение к syslogd неавторизованных серверов.

Первоначально syslogd принимал зарегистрированные сообщения от любого сервера в Сети, однако при этом кто-нибудь мог заполнить этими сообщениями весь ваш жесткий диск. Чтобы разрешить тому или иному хосту отправлять сообщения демону syslogd, укажите этот хост с ключом `-a`. Для этого задайте его IP-адрес с сетевой маской, определенной как количество битов (см. главу 5).

Так, чтобы разрешить хосту принимать зарегистрированные сообщения с любого хоста с адресом в диапазоне от 192.168.0.0 до 192.168.0.255, примените такую строку:

```
syslogd_flags="-s -a 192.168.1.0/24"
```

Также можно указать имена хостов:

```
syslogd_flags="-s -a mail.absolutebsd.org"
```

Запись имени хоста опирается на обратный DNS. Если на какой-нибудь системе в другой сети изменили запись обратного DNS, чтобы она соответствовала разрешенному хосту, то с этой системы к вашему syslogd будут приходить якобы достоверные сообщения. Лучше применять IP-адреса.

`inetd_enable="NO"`

Данный параметр блокирует сервер inetd (см. главу 13). Если необходимо запускать inetd, присвойте этой опции значение `"YES"`.

`named_enable="NO"`

Поскольку для надлежащей работы программы named ее необходимо конфигурировать, во FreeBSD она по умолчанию заблокирована (см. главу 13). Для запуска named присвойте этому параметру значение `"YES"`.

`named_program="named"`

Если вы собрали свою программу named, укажите здесь полный путь к ней. В коллекции «портов» есть несколько специализированных серверов имен (см. главу 10).

`named_flags="-u bind -g bind"`

Задает флаги для named(8). Значения по умолчанию приемлемы, однако в этом поле можно разместить любые допустимые опции. Программа named подробно рассмотрена в главе 12.

`sshd_enable="NO"`

Активизирует или отключает демон ssh, обеспечивающий безопасное подключение к системе по сети. Конфигурирование sshd подробно описано в главе 13.

`sshd_flags=""`

Демон SSH можно конфигурировать с помощью флагов в командной строке, однако для тонкой настройки сервиса ssh лучше отредактировать файл */etc/ssh/sshd_config*.

```
ntpdate_enable="NO"
ntpdate_flags="-b"
xntpd_enable="NO"
xntpd_flags="-p /var/run/ntpd.pid"
```

Эти четыре переменных управляют демонами сетевого хронометража. Подробно эти программы рассмотрены в главе 13.

Параметры сетевой маршрутизации

Во FreeBSD параметры маршрутизации настраиваются отдельно от основных сетевых параметров.

`defaultrouter=""`

Указывает IP-адрес шлюза сети.

`gateway_enable="NO"`

Установите значение `"YES"`, если в системе есть несколько сетевых интерфейсов, через которые необходимо передавать пакеты из одной сети в другую. Пропуская трафик от одного интерфейса к другому, система становится шлюзом.

`router_enable="NO"`

Если система является шлюзом, а таблицу маршрутизации необходимо поддерживать с помощью протокола RIP, присвойте параметру значение `"YES"`. В противном случае держитесь от нее подальше!

Параметры консоли

Параметры консоли управляют монитором и клавиатурой. С их помощью можно изменить язык клавиатуры, размер шрифта на мониторе и многое-многое другое.

`keymap="NO"`

С помощью этого параметра можно указать системе на необходимость использования другой раскладки клавиатуры. В */usr/share/syscons/keymaps* есть довольно много файлов раскладок с различными схемами расположения для различных стран.

Я использую раскладку Dvorak, которую довольно легко установить с помощью такой строки:

```
keymap="us.dvorak.kbd"
```

`blanktime="300"`

Поле `blanktime` определяет, сколько секунд клавиатура должна бездействовать, прежде чем FreeBSD предпишет монитору перейти в режим энерго-

сбережения (в данном случае пройдет 300 секунд). Если установить значение "NO", FreeBSD не будет гасить экран.

Примечание

Некоторые современные аппаратные средства поддерживают автоматизированные функции гашения экрана. Если монитор настаивает на переходе в режим бездействия, а этому параметру присвоено значение NO", проверьте BIOS и руководство по монитору.

`moused_enable="NO"`

Активизируйте этот параметр для использования мыши на консоли. Консольная мышь позволит выделять, копировать и вставлять фрагменты текста.

`moused_type="auto"`

Для преобразования движения колесика мыши в перемещения указателя могут применяться различные протоколы. Обычно FreeBSD довольно хорошо справляется с автоматическим обнаружением протокола, который использует та или иная мышь. Однако если автообнаружение не сработало, эту опцию можно установить вручную. Допустимые значения опции представлены на странице руководства moused(8).

`moused_port="/dev/psm0"`

Задает физический порт, к которому подключена мышь. По умолчанию это /dev/psm0 – порт для мыши PS/2. Если к последовательному порту 0 (com1) подключена последовательная мышь, присвойте параметру значение /dev/cuaa0. Второй последовательный порт – /dev/cuaa1.

`moused_flags=""`

Демон мыши можно эффективно настраивать для поддержки разнообразных мышей, которые когда-либо подключались к машинам FreeBSD. Сегодня большинство мышей – либо USB, либо PS/2, поэтому данные параметры обычно бесполезны. В moused(8) подробно описано, как заставить правильно работать шаровой манипулятор, который требует 1350 бод.

Параметр, по-прежнему распространенный в настоящее время, равен -3. В обычной мыши UNIX есть от трех до пяти кнопок. Многие программы UNIX подразумевают наличие третьей кнопки мыши. Данный флаг позволяет эмулировать эту кнопку одновременным нажатием двух других.

Прочие параметры

Наконец, есть несколько параметров, не подпадающих под все остальные категории.

`lpd_enable="NO".`

Присвойте этому параметру значение "YES", если с этой системы необходимо выполнять печать. Основы печати подробно рассмотрены в главе 21.

`usbd_enable="NO"`

При наличии устройств USB активизируйте этот параметр.

`sendmail_enable="NO"`

Позволяет системе получать почту от других систем. Ее следует активизировать лишь на почтовых серверах.

`sendmail_outbound_enable="YES"`

Позволяет системе посылать почту другим компьютерам. Это необходимо почти на всех системах.

`dumpdev="NO"`

Для сохранения образа ядра при аварии (для отладки) присвойте этому параметру имя раздела свопинга. Размер этого раздела должен быть больше емкости физической памяти или равен ей. Название раздела свопинга можно найти в */etc/fstab*. На моем лэптопе это */dev/ad0s1b*, а параметр я установил так:

```
dumpdev="/dev/ad0s1b"
```

В главе 20 подробно описано применение `dumpdev` при аварии системы.

`ibcs2_enable="NO"`

Включает или выключает совместимость ядра с SCO UNIX. См. главу 11.

`linux_enable="NO"`

Включает или выключает модуль, обеспечивающий совместимость ядра с Linix. См. главу 11.

`svr4_enable="NO"`

В состав FreeBSD входит модуль, обеспечивающий совместимость с UNIX System V. Данный параметр включает его или отключает. Подробности см. в главе 11.

`osf1_enable="NO"`

DEC Digital UNIX работает только на Alpha. Поскольку FreeBSD также может работать на Alpha, существует модуль для обеспечения совместимости с этой системой. Этот параметр включает его или отключает. (Для аппаратных средств X86 он бесполезен и будет обсуждаться только теоретически, но о его существовании полезно знать.)

`clear_tmp_enable="NO"`

В старых системах UNIX содержимое */tmp* уничтожается при начальной загрузке. Во FreeBSD это не выполняется по умолчанию, однако здесь эту операцию можно активизировать.

`ldconfig_paths="/usr/lib/compat /usr/X11R6/lib /usr/local/lib"`

В этом параметре перечисляются каталоги, в которых хранятся разделяемые библиотеки. В большинстве систем пригодно значение по умолчанию. Однако если необходимо указать другие каталоги, то вместо того, чтобы устанавливать `LD_LIBRARY_PATH` для всех пользователей, лучше настроить этот параметр. В главе 11 есть и другие советы.

`kern_securelevel_enable="NO"`

Если присвоить этому параметру значение `"YES"`, то при загрузке будут активизироваться функции защиты, реализованные в ядре FreeBSD. См. главу 7.

`kern_securelevel="-1"`

Если активизирована защита ядра, с помощью этого параметра можно задать уровень безопасности. См. главу 7.

`start_vinum="NO"`

Присвойте этому параметру значение `"YES"`, если в системе применяется RAID на основе программного обеспечения vinum. См. главу 17.

rc.shutdown

При выполнении команды shutdown или reboot система запустит *rc.shutdown*. Этот сценарий ищет локальные стартовые каталоги, указанные в */etc/rc.conf*, и запускает каждый найденный сценарий с аргументом `"stop"`.

Если при останове системы необходимо выполнять определенные действия, в конец файла */etc/rc.shutdown* можно поместить соответствующие команды оболочки. В большинстве программных пакетов соответствующие команды останова включены в сценарий */usr/local/etc/rc.d*, поэтому добавлять их не надо. Однако сюда можно добавить собственные команды. Для этого желательно создать новый сценарий и разместить его в стартовом каталоге.

/etc/resolv.conf

Файл */etc/resolv.conf* управляет работой распознавателя DNS. См. главу 12.

/etc/security

Файл */etc/security* содержит сценарий оболочки, запускаемый ежедневно программой periodic(8). Его можно редактировать по своему усмотрению. Этот сценарий выполняет различные проверки целостности системы. Например, проверяет изменения в */etc/master.passwd*, монтированных файловых системах, а также протоколы работы ядра. Результаты проверки ежедневно посылаются пользователю root. Чтобы полностью отключить эту проверку, установите в файле */etc/periodic.conf* такое значение:

```
daily_status_security_enable="NO"
```

Обычно проверка защиты оправдывает себя, поскольку помогает обнаружить неполадки в системе и бреши в защите.

/etc/services

В этом файле перечислены применяемые сетевые порты. См. главу 5.

/etc/shells

/etc/shells содержит список командных процессоров, доступных пользователю. При установке командного процессора из «порта» или «пакета» в */etc/shells* будет добавлена соответствующая запись. Однако если не применять «порт», а компилировать исходный код, то этот файл надо отредактировать. Командные процессоры приведены вместе с путями к ним (path names)

Демон FTP отклонит подключение пользователя, командный процессор которого не перечислен в */etc/shells*. Если */sbin/nologin* используется как командный процессор исключительно для доступа по FTP, то эту программу надо добавить в */etc/shells*. Однако более правильный путь состоит в том, чтобы создать собственный login-класс (см. /etc/login.conf).

/etc/spwd.db

Этот файл похож на */etc/pwd.db*, но основан на */etc/master.passwd*. Он содержит информацию обо всех учетных записях пользователей в виде базы данных, поэтому другие программы могут быстро к ним обратиться. Поскольку в этом файле находится конфиденциальная информация, его может читать только root. Для получения подробностей обратитесь к */etc/pwd.db* и */etc/master.passwd*.

/etc/ssh

Этот файл управляет работой сервера и клиента SSH. За более подробной информацией обратитесь к главе 13.

/etc/sysctl.conf

Этот файл содержит информацию о том, какие sysctls ядра устанавливаются в процессе начальной загрузки. См. главу 4.

/etc/syslog.conf

Этот файл определяет, какие данные будут протоколироваться системой. См. главу 19.

10

Делаем систему полезной

В таких операционных системах, как Microsoft Windows и Red Hat Linux, базовая установка подразумевает инсталляцию всех компонентов, которые когда-либо могут понадобиться пользователю. В системах BSD это не так: компоненты устанавливаются разрозненно, и в этом есть свое преимущество.

Например, система Windows 2000 Professional, которую я использую на клиентском сайте, при «минимальной» установке создает 1768 файлов в основном системном каталоге (*C:\WINNT\system32*) и размещает на диске чуть ли не все разделяемые библиотеки, известные как библиотеки динамической компоновки (Dynamic Link Libraries, DLL). Когда бы система ни запускалась, эти библиотеки загружаются в системную память. Я не знаю, для чего предназначена каждая библиотека, но могу гарантировать, что никогда не буду использовать многие из них. Единственные программы, которые я применяю на этой машине, это SSH и Mozilla. Все остальные программы только поглощают оперативную память.

Да, именно так Microsoft разрабатывает операционные системы – пользователям надо дать все, что только у вас есть, то есть все от первой до последней мелочи. Наоборот, при установке Red Hat Linux инсталлируется сравнимый объем программного обеспечения, но его бульшую часть составляют полезные программы. Возможно, многие из них вам никогда не понадобятся, но все эти файлы, по крайней мере, не будут автоматически размещаться в оперативной памяти при начальной загрузке.

При установке BSD инсталлируется столько программ, сколько требуется для запуска системы, плюс дополнительные утилиты, традиционно включаемые в состав систем UNIX. Программа установки предлагает выбор: инсталлировать дополнительные программы и исходный код или нет. Однако даже полная система BSD занимает намного меньше дискового пространства, чем каталог *\system32* системы Windows 2000, упомянутый ранее. По сравнению с Windows, при полной установке FreeBSD инсталлируется намного меньше компонентов. Система Windows, поддерживающая только пакеты SSH и Mozilla, была бы намного меньше и проще – совсем как FreeBSD.

Преимущество такой разбивки заключается в том, что можно устанавливать лишь необходимые компоненты системы. За счет этого существенно упро-

щается поиск и устранение неполадок, а неизвестные разделяемые библиотеки, которые никогда не понадобятся, не вызовут сбои в системе. Обратная сторона медали: необходимо поразмыслить о том, что именно вам требуется, а затем установить дополнительные программы. Однако разработчики FreeBSD позаботились о том, чтобы установка программного обеспечения была настолько простой, насколько возможно.

Сборка программного обеспечения

Сборка программного обеспечения представляет собой сложный процесс, поскольку исходным кодом необходимо распорядиться так, чтобы создать работающие исполняемые файлы, – не говоря о том, что они должны быть оптимизированы!

Программисты могли бы включать в свои программы подробные инструкции по установке, например «Теперь наберите `cc CPUTYPE=i686 -ohttpd -I /usr/src/crypto/kerberosIV/include -lcrypto -lkrb`», но они этого не делают. Их терпения хватает ненадолго. Если эта операция автоматизирована, то в программах будет документация, что очень помогает при их установке.

Основной инструмент для сборки программного обеспечения – утилита make(1). В текущем каталоге она ищет файл с именем Makefile, который изобилует ужасными командами, подобными примеру из предыдущего абзаца. Когда Makefile найден, программа make читает команды и выполняет их. Обычно Makefile имеет большой размер и труден для восприятия. На самом деле знать начинку Makefile вовсе не обязательно, поэтому здесь он не анализируется.

Каждый Makefile включает в себя различные *целевые объекты* (*targets*), или типы исполняемых команд. Например, `make install` ищет в Makefile целевой объект (процедуру) с именем «install». Если такая процедура найдена, она выполняется. Каждый целевой объект представляет собой один базовый этап в создании, установке и конфигурировании программного обеспечения. Различные команды make targets будут обсуждаться как в этой главе, так и далее по мере их использования.

make может выполнять огромное количество функций, многие из которых превосходят первоначальные намерения создателей. Но именно для этого и предназначена система UNIX, не правда ли?

Примечание

При запуске make убедитесь, что текущим является каталог, в котором находится Makefile. Хотя это не строгое правило, оно упростит работу с make.

Горе и радости исходного кода

Исходный код – это удобочитаемые команды для создания фактического машинного кода, который образует программу. Возможно, вы уже открыли для себя исходный код в той или иной форме. Если же вы никогда его не видели, взгляните на различные файлы в каталоге */usr/src*.

Уметь читать исходный код вовсе не обязательно, но уметь распознавать его в двух случаях из трех необходимо. Вот фрагмент исходного кода из сетевого стека FreeBSD:

```
/* While we overlap succeeding segments trim them or,
* if they are completely covered, dequeue them.
*/
/* Если последовательно идущие сегменты частично перекрываются, подвергнем их
* усечению или, если они перекрываются полностью, удалим их из очереди. */
while (q) {
  register int i = (th->th_seq + *tlenp) - q->tqe_th->th_seq;
        if (i <= 0)
                break;
        if (i < q->tqe_len) {
                q->tqe_th->th_seq += i;
                q->tqe_len -= i;
                m_adj(q->tqe_m, i);
                break;
        }
```

При наличии исходного кода программы установить ее достаточно просто. Надо собрать (или *скомпилировать*) программу в системе, в которой она будет выполняться.[1] Если программа была написана для операционной системы, в значительной степени подобной платформе, на которой будет запускаться программа, то компиляция будет успешной. Если же целевая платформа существенно отличается от оригинальной, программа не будет скомпилирована. После успешной сборки программного обеспечения на целевой платформе получившуюся программу (т. е. **исполняемый файл**, **binary**) можно копировать на другие идентичные платформы – она должна работать и там.

Некоторые программы написаны настолько хорошо, что их можно компилировать на множестве различных платформ. В отдельные программы специально включена поддержка для платформ, значительно отличающихся друг от друга. Например, веб-сервер Apache можно компилировать как в Windows, так и в UNIX – для этого достаточно набрать `make install`. Впрочем, это нетипичный случай, который является свидетельством поистине героических усилий разработчиков программного обеспечения.

Примечание

Попытка копирования скомпилированной программы на другую платформу и запуска ее там, как правило, обречена на провал. В большинстве случаев в одной операционной системе нельзя так вот запросто запускать программы, собранные для другой системы. (Это возможно во FreeBSD, однако для этого необходимо дополнительно настроить систему; см. главу 11.)

[1] Собирать программное обеспечение можно и на иной платформе с помощью так называемой «кросс-компиляции». Однако для этого недостаточно знаний, полученных при изучении материала этой книги.

Отладка

Вообще говоря, если удастся собрать программу из исходного кода, то она будет выполняться в вашей системе UNIX; а если не удастся, то программа не будет работать. Впрочем, умудренный опытом администратор, изучив исходный код, может сказать, почему программа не будет собрана или выполнена. В большинстве случаев проблема проста, и обойти ее можно с минимальными усилиями. (Это одна из причин, объясняющая важность доступа к исходному коду.)

Раньше каждый администратор UNIX был программистом, а отладка поглощала значительную часть его времени. Каждая система UNIX незначительно отличалась от других, поэтому все администраторы должны были понимать платформу, для которой программа была написана, и ее отличия от своей платформы. Лишь в этом случае можно было рассчитывать на успешный запуск данной программы. Приходилось затрачивать двойные усилия и это было поистине чудовищно.

Со временем были созданы инструменты autoconf и configure, предназначавшиеся для облегчения кросс-компилирования. Тем не менее не каждая программа использовала эти инструменты, поэтому при возникновении сбоев администратор начинал с самого начала. Системным администраторам приходилось редактировать исходный код и файлы Makefile, чтобы обечпечить хотя бы вероятность надлежащей работы программ.

Система «портов» и пакетов

Система «портов» и пакетов FreeBSD – это система сборки программного обеспечения, разработанная для упрощения конфигурирования и установки программ. Начало применению этой системы было положено в 1995 году.

«*Порты*» (*ports*) – это команды для компиляции программ во FreeBSD, а пакеты – это просто предварительно скомпилированные «порты». Пакеты можно установить быстрее, что позволяет сэкономить время. «Порты» устанавливаются медленнее, но принимают изменения от операционной среды (они указываются в */etc/make.conf*).

Основная идея системы «портов» и пакетов очень проста: если программу необходимо модифицировать для запуска в BSD, то эти модификации надо автоматизировать. Автоматизируя эти изменения, можно позаботиться об учете всех файлов, входящих в состав программы. Тогда программу можно будет легко инсталлировать и деинсталлировать. Так как процесс сборки программного обеспечения каждый раз выдает один и тот же результат, а все вновь создаваемые файлы известны, то можно скопировать все исполняемые файлы и установить их на любой похожей системе FreeBSD. На этом основана система «портов» и пакетов.

Вся система называется *коллекцией «портов»* (*ports collection*), *деревом «портов»* (*ports tree*) и даже просто «*портами*» (*ports*). Когда кто-нибудь

употребляет один из этих терминов, он обычно подразумевает «порты», систему для создания «портов» и пакеты.

«Порты»

«Порт» – это набор команд, задающих правила наложения *заплаток* (*patches*) на файлы исходного кода. Комбинируя заплатки и команды установки, FreeBSD может вести полный учет всего процесса установки программного обеспечения. Такой подход избавляет от трудностей установки программ и позволяет сконцентрироваться на их надлежащей работе.

Если вы следовали инструкциям по установке, представленным в главе 1, то дерево «портов» установлено в */usr/ports*. Листинг выглядит примерно так:

```
# ls /usr/ports/
INDEX          cad            games          misc           ukrainian
LEGAL          chinese        german         net            vietnamese
Makefile       comms          graphics       news           www
Mk             converters     hebrew         palm           x11
README         databases      irc            picobsd        x11-clocks
Templates      deskutils      japanese       print          x11-fm
Tools          devel          java           russian        x11-fonts
archivers      distfiles      korean         science        x11-servers
astro          editors        lang           security       x11-toolkits
audio          emulators      mail           shells         x11-wm
benchmarks     french         math           sysutils
biology        ftp            mbone          textproc
#
```

Если в */usr/ports* нет подобного содержимого, то для продолжения работы необходимо установить «порты». Для этого зайдите на ближайший FTP-сервер FreeBSD и найдите каталог, соответствующий вашей версии FreeBSD. В нем есть каталог *ports*. Поищите два файла, *ports.tgz* и *install.sh*; загрузите оба и запустите install.sh. После выполнения этой команды вы должны увидеть что-либо подобное предыдущему листингу.

Каталоги, показанные в предыдущем листинге, представляют собой категории программного обеспечения. В каждой категории есть вложенный уровень каталогов, каждый из которых и есть «порт» той или иной программы. На момент написания этих строк в состав FreeBSD входило почти 6000 «портов», поэтому очень важно поддерживать дерево каталогов в каком-либо определенном порядке!

Следующий листинг показывает содержимое категории «astro», где хранится программное обеспечение для астрономических исследований, которое поддерживается системой FreeBSD. (Да, система FreeBSD применяется для выполнения серьезных астрономических задач.) Для большинства пользователей эта категория не представляет особого интереса, однако у нее есть важное преимущество: она достаточно мала, чтобы ее можно было описать в книге. Другие категории «портов», например «www», насчитывают сотни файлов.

```
#ls /usr/ports/astro/
Makefile                p5-Astro-SunTime        sunclock
SETIsupport             p5-Astro-Sunrise        tkseti
dgpsip                  p5-GPS                  wmglobe
ephem                   p5-Geo-METAR            wmmoonclock
fooseti                 pkg                     wmspaceweather
gkrellmearth            pyweather               wmsun
gkrellmoon              rmap                    x3arth
glunarclock             saoimage                xearth
jday                    sattrack                xephem
ksetiwatch              seti_applet             xglobe
luna                    setiathome              xphoon
openuniverse            sscalc                  xplanet
p5-Astro-MoonPhase      stars                   xtide
#
```

Поиск программного обеспечения

Некоторые категории содержат сотни «портов». Как же найти в них что-либо? Файл */usr/ports/INDEX* содержит список всех «портов» в алфавитном порядке. Это индекс «портов», в котором каждый «порт» описан в отдельной строке, а поля разделены символами вертикальной черты «|».

Хотя такой формат очень удобен для различных системных инструментов, он не особенно удобочитаем. Для чтения этого файла необходимо знать смысл каждого поля. (Некоторые из них кажутся избыточными, но они необходимы по причинам, которые будут изложены позднее.) Вот простая запись с обрывом строки на каждом разделителе – для облегчения восприятия:

```
❶ fooseti-0.6.5|
❷ /usr/ports/astro/fooseti|
❸ /usr/local|
❹ GTK+ frontend to SETI@Home|
❺ /usr/ports/astro/fooseti/pkg-descr|
❻ petef@databits.net|
❼ astro|
❽ XFree86-3.3.6_9 gettext-0.10.35 glib-1.2.10_3 gtk-1.2.10_2|
❾ XFree86-3.3.6_9 gettext-0.10.35 glib-1.2.10_3 gtk-1.2.10_2|
❿ http://www-personal.engin.umich.edu/~agorski/fooseti
```

Первое поле (❶) – имя и номер версии программного пакета. В нашем случае это fooseti версии 0.6.5. Второе поле (❷) – каталог, в котором можно найти этот «порт» (*/usr/ports/astro/fooseti*), а следующее поле (❸) – место установки по умолчанию. К примеру, «порт» fooseti устаналивается в */usr/local*, если администратор не изберет другое расположение. Далее следует краткое описание программного пакета (❹).

Пятое поле (❺) задает местоположение файла с более полным описанием продукта. Поле электронного адреса (❻) сообщает адрес разработчика FreeBSD, который отвечает за поддержку надлежащей работы этого «порта». Следующее поле – это категория (❼), а именно каталог */usr/ports*, в котором находится каталог «порта».

Восьмое поле (❽) содержит список «портов», необходимых для сборки этой программы. Для многих «портов» обязательно наличие других «портов»; например, программе может понадобиться специальная версия make, называемая[1] *зависимостью сборки* (*build dependency*). В нашем примере необходимы «порты» XFree86, gettext, glib и gtk.

Девятое поле (❾) перечисляет «порты», необходимые для запуска этой программы. В дополнение к зависимостям сборки многие «порты» имеют *зависимости запуска* (*runtime dependencies*). Это означает, что при выполнении программы вызываются другие программы. Если зависимости не найдены, программа не может быть выполнена. В данном примере зависимости сборки и запуска идентичны, хотя так бывает не всегда.

Последнее поле – это URL домашней страницы программы (❿).

Примечание

Если вы забыли назначение каждого из полей, то с помощью команды `make print-index` выведите более полный список всего содержимого индекса.

Поиск по имени

Знать содержимое индекса – это прекрасно, но как с его помощью найти ту или иную программу?

Если точное имя программного пакета известно, на помощь придет простая команда grep. Это быстрый и легкий путь, если вы уверенно работаете с grep(1) *и* знаете точное имя программы в дереве «портов» FreeBSD. Например, для того чтобы найти пакет staroffice, введите такую команду grep:

```
# grep -i ^staroffice INDEX
staroffice-5.1a|/usr/ports/editors/staroffice5|/usr/local|Integrated wordprocessor/
dbase/spreadheet/drawing/chart/browser|/usr/ports/editors/staroffice5/
pkg-descr|mb@imp.ch|editors linux|unzip-5.42|linux_base-6.1|staroffice-5.2|
/usr/ports/editors/staroffice52|/usr/local|Integrated wordprocessor/dbase/
spreadheet/drawing/chart/browser|/usr/ports/editors/staroffice52/
pkg-descr|mb@imp.ch|editors linux|linux_base-6.1||
#
```

Вывод grep показывает наличие двух различных версий staroffice, версии 5.1a и версии 5.2. Обе версии доступны в */usr/ports/editors* – в каталогах *staroffice5* и *staroffice52* соответственно.

[1] Имеется в виду, что в каком-то конкретном случае может понадобиться такая версия make. Вообще-то любая программа, необходимая для установки другой программы, называется dependency. Устоявшегося перевода этот термин не имеет, т. к. о программе, зависящей от какой-то другой программы, можно сказать «зависимая», а ту, от которой она зависит, приходится называть «зависимостью». – *Примеч. науч. ред.*

Поиск по частичному имени

Если точное имя программы неизвестно, попробуйте задействовать функцию поиска, предоставляемую коллекцией «портов». Команда `make search` просканирует весь индекс «портов», разыскивая либо имя «порта», либо «порты», в которых встречается это слово.

Например, если вы ищете популярный менеджер файлов Midnight Commander, попробуйте выполнить такую команду:

```
# make search name=midnight
#
```

Да, полезного мало.

Поиск по ключевому слову

Если такой поиск ничего не дал (как в предыдущем примере), обратитесь к более общему поиску, задав параметр `key`. В этом случае будет просканировано больше полей и возвращено больше совпадений. (Однако если искать обычное слово, то по ключевому слову можно получить слишком много информации.)

Вот одна из возможностей поиска по ключевому слову «midnight»:

```
# make search key=midnight
Port:   mc-4.5.54_2
Path: ? /usr/ports/misc/mc
Info:   Midnight Commander, a free Norton Commander Clone
Maint:  gnome@FreeBSD.org
Index:  misc
B-deps: gettext-0.10.35 glib-1.2.10_3 gmake-3.79.1
R-deps: gettext-0.10.35 glib-1.2.10_3
#
```

Ага! Midnight Commander можно найти в *? /usr/ports/misc/mc*.

Другие способы просмотра коллекции «портов»

Если вы предпочитаете работать с веб-броузером, соберите индекс в формате HTML. Перейдите в */usr/ports* и (как root) наберите `make readmes` для создания файла с содержимым дерева «портов» (*README.html*). (В нем можно щелкать мышью по различным категориям и даже просматривать подробное описание каждого «порта».)

Если ни один из этих вариантов не дал результатов, попробуйте обратиться к сервису поиска FreeBSD Ports Tree по адресу *http://www.FreeBSD.org/cgi/ports.cgi*.

Применив веб-броузер и поисковую машину, вы сможете найти необходимую программу.

Правовые ограничения

В то время как большинство программ в коллекции «портов» открыты для некоммерческого использования, некоторые из них имеют необычные пра-

вовые ограничения. В файле */usr/ports/LEGAL* перечислены правовые ограничения для различных программ. Наиболее типичное ограничение – запрет на передачу третьим лицам. Проект FreeBSD не включает такие программы в дистрибутивы, поставляемые на компакт-дисках, а предоставляет только инструкции по их сборке. Например, довольно долго FreeBSD не имела лицензии на Java. Проекту возбранялось распространять исходный код Java или скомпилированные исполняемые файлы. Впрочем, можно было распространять инструкции по сборке программ из исходного кода. Вы могли зайти на веб-страницу Sun Microsystems, скачать исходный код Java и собрать свою версию Java на FreeBSD. Аналогично для некоторых программ разрешено коммерческое использование либо включение в состав коммерческих продуктов. Некоторые программы нельзя экспортировать из Соединенных Штатов вследствие Правил международной торговли оружием (International Traffic in Arms Regulations, ITAR) – в этих программах применяется криптография, и они классифицируются как «снаряжение».[1] Если вы собираете системы FreeBSD для последующей продажи (раздачи), экспорта или коммерческого использования, вам определенно следует изучить этот файл.

К счастью, программное обеспечение, необходимое для организации сетевых служб, открыто как для коммерческого, так и некоммерческого использования. Пакеты с ограничениями – это исключение из правила.

Применение пакетов

Пакеты – это предварительно скомпилированные программы для определенной версии FreeBSD. Сначала мы обсудим именно пакеты, поскольку, как правило, работать с ними легче, чем с «портами». После рассмотрения пакетов обратимся к «портам».

Если на распространение программы в компилированном виде не наложены правовые ограничения, то она доступна как пакет. Другие программы (например, Microsoft Word)[2] доступны только в компилированном виде. Пакеты доступны на компакт-дисках или через FTP.

Установка программы как пакета может сэкономить много времени, потому что оно не будет тратиться на компиляцию исходного кода. Как было описано ранее, для установки пакета надо найти его имя в дереве «портов».

---

[1] Большинство этих программ доступны из источников за пределами США – их можно скачать откуда угодно. Однако официальные образы FreeBSD, размещаемые на компакт-дисках, производятся в Соединенных Штатах.

[2] Нет, программа Microsoft Word недоступна во FreeBSD. Пока. Однако очень трудно привести значительный, представительный пример программы для BSD, поставляемой только в виде исполняемого файла. Почти все программы доступны в исходном коде.

Установка с CD-ROM

В комплекте компакт-дисков FreeBSD есть довольно значительная коллекция скомпилированных пакетов. Для их использования нужно лишь смонтировать CD и прочитать файл с описанием пакета.

Монтирование и демонтирование носителей будут подробно рассмотрены в главе 16. Здесь представлены только основы.

Вставьте компакт-диск в привод и (как пользователь root) наберите такую команду:

```
# mount -t cd9660 /dev/acd0c /cdrom
#
```

Теперь содержимое компакт-диска доступно в */cdrom*.

Примечание

Вы не сможете извлечь компакт-диск, если обращаетесь к нему или если он смонтирован.[1] Если текущий каталог – это */cdrom*, демонтировать компакт-диск не удастся. Для демонтирования введите такую команду:

```
# umount /cdrom
```

После монтирования CD взгляните на каталог *packages*:

```
# cd /cdrom/packages/
# ls
All             deskutils       japanese        print           tk82
INDEX           devel           java            python          tk83
Latest          editors         kde             ruby            tkstep80
archivers       elisp           korean          russian         windowmaker
astro           emulators       lang            security        www
audio           french          mail            shells          x11
biology         ftp             math            sysutils        x11-fm
cad             games           mbone           tcl80           x11-fonts
chinese         german          misc            tcl82           x11-toolkits
comms           gnome           net             tcl83           x11-wm
converters      graphics        palm            textproc        zope
databases       ipv6            perl5           tk80
#
```

Листинг кажется знакомым. Да, он такой же, как и листинг дерева «портов», рассмотренный ранее в этой главе. Однако если войти в каталог, можно увидеть незначительные различия. На одном компакт-диске недостаточно места для хранения всех пакетов FreeBSD, объем которых может быть довольно велик (сотни мегабайт). Здесь представлены некоторые пакеты, но не все. Многие пакеты доступны на других компакт-дисках FreeBSD. Второе от-

[1] Точнее, если хоть кто-нибудь обращается к компакт-диску или если каталог (или его подкаталог), куда смонтирован диск, является текущим каталогом какого-то пользователя. – *Примеч. науч. ред.*

личие: это файлы, а не каталоги. Они представляют собой архивы (tarballs), содержащие полные программные пакеты.

Например, в */cdrom/packages/astro* есть два пакета. Оба они основаны на «портах», которые можно найти в каталоге *astro* дерева «портов».

```
# cd astro/
# ls
openuniverse-1.0.b3.tgz xglobe-0.5.tgz
#
```

О назначении пакета можно узнать по его описанию в /usr/ports/INDEX. Поищите имя пакета в индексном файле (аналогично поиску имени «порта» в индексе, см. раздел «Поиск программного обеспечения» ранее в этой главе). В четвертом поле представлено описание «порта»:

```
# grep -i ^openuniv /usr/ports/INDEX
openuniverse-1.0.b3|/usr/ports/astro/openuniverse|/usr/X11R6|OpenGL Solar System
simulator for X Window System|/usr/ports/astro/openuniverse/pkg-descr|
trevor@FreeBSD.org|astro|Mesa-3.4.2_1 XFree86-libraries-4.1.0 freetype2-2.0.4
gettext-0.10.35 gmake-3.79.1 imake-4.1.0 jpeg-6b|Mesa-3.4.2_1 XFree86-libraries-
4.1.0 freetype2-2.0.4 imake-4.1.0 jpeg-6b|http://openuniverse.sourceforge.net/
#
```

Четвертое поле описания сообщает, что openuniverse – это «OpenGL Solar System simulator for X Window System» (Имитатор Солнечной системы для X Window System, использующий OpenGL). Имитатор солнечной системы – звучит классно, правда? Давайте его установим! Для установки пакетов применяется pkg_add(1):

```
# pkg_add openuniverse-1.0.b3.tgz
#
```

Вот так! Программа установлена и готова к выполнению. (Обычно установка проходит «молча», однако изредка на экране появляются сообщения. Обращайте на них внимание и выполняйте действия, которые они рекомендуют.)

Если пакету необходимы другие пакеты, pkg_add(1) автоматически найдет их и установит. Набор компакт-дисков спроектирован так, что зависимые программы (dependencies) по возможности располагаются на одном диске. Однако если требуемого пакета нет, pkg_add выдаст название отсутствующего пакета и остановится. В этом случае найдите требуемый пакет на другом диске и установите его. Другой вариант состоит в том, чтобы установить его по FTP.

Установка через FTP

Зачастую пакет не представлен на CD, потому что пространство в наборе из 4 компакт-дисков Проекта FreeBSD ограничено и не может вместить больше 6000 пакетов! Кроме того, программное обеспечение на CD собрано для определенного «выпуска» FreeBSD. Наличие компакт-диска с пакетами для версии 4.4 не поможет при запуске FreeBSD 4.6.

Кроме того, в случае -stable пакеты на CD слегка устаревают, поэтому приходится загружать последнюю версию пакета с *ftp.FreeBSD.org*. (Для этого необходимо устойчивое соединение с Интернетом!)

Если известны полное название пакета и номер версии, свежую версию пакета можно скачать с FTP-сайта FreeBSD автоматически:

```
# pkg_add -r xearth
Fetching ftp://ftp.FreeBSD.org/pub/FreeBSD/ports/i386/packages-4.4-release/Latest/
xearth.tgz... Done.
#
```

Преимущество такого подхода: система автоматически найдет подходящий сервер FTP, загрузит подходящую версию пакета и все программы, от которых он зависит, а затем установит их. Обратная сторона медали: необходимо устойчивое соединение с Интернетом.

Кроме того, этот метод менее безопасен, нежели установка с CD. Все пакеты, размещаемые на CD, проверяются и верифицируются на соответствие описанию, а пакеты на сервере FTP могут быть осквернены каким-нибудь злоумышленным хакером. Можно ненароком установить троянских коней или еще что похуже. (Уверяю вас, такое никогда не случалось, но теоретически это возможно.)

Пакеты можно также скачать вручную с сайта FTP по своему выбору. (Поиск удобного сервера FTP рассмотрен в главе 1.) Для этого найдите подходящий сайт FTP и зарегистрируйтесь на этом сервере. Далее, если вы работаете с -release, перейдите в *pub/FreeBSD/release*, а затем в каталог, соответствующий вашей версии -release. В случае -stable или -current перейдите в *pub/FreeBSD/ports* и выберите каталог, соответствующий вашей версии -stable или -release.

Оказавшись в подходящем каталоге, можно увидеть дерево каталогов, в значительной степени подобное дереву в */usr/ports*. Осталось найти требуемый пакет, загрузить его и установить из командной строки:

```
# pkg_add openuniverse-1.0.b3.tgz
#
```

Примечание

При выборе этого метода зависимости не устанавливаются автоматически. Такой метод наиболее полезен в тех случаях, когда вы находитесь за брандмауэром и должны приложить дополнительные усилия для загрузки файлов по Интернету.

Что устанавливает пакет?

Теперь, когда программа установлена, как найти ее в системе? В конце концов, здесь нет меню Start! Беспокоиться не о чем.

Для получения полного списка установленных программ обратитесь к */var/db/pkg*. В этом каталоге представлены все «порты» или пакеты, установленные в системе, а также их содержимое.

К примеру, наш */var/db/pkg* сейчас содержит каталог, называемый *openuniverse-1.0.b3*. В нем находятся следующие файлы:

```
# ls /var/db/pkg/openuniverse-1.0.b3/
+COMMENT        +CONTENTS       +DESC
#
```

Файл *+COMMENT* содержит краткое описание пакета, а файл *+DESC* – более подробное описание. Особый интерес представляет файл *+CONTENTS*, в котором перечислен каждый файл, установленный пакетом. Этот файл довольно большой, и здесь мы ознакомимся с его началом.

```
# more /var/db/pkg/openuniverse-1.0.3b/+CONTENTS
❶ @name openuniverse-1.0.b3
@cwd ❷ /usr/X11R6
❸ @pkgdep jpeg-6b
@pkgdep Mesa-3.4.1
@comment ❹ ORIGIN:astro/openuniverse
❺ bin/openuniverse
@comment MD5:2a4775c079a589e78cf54be5444316cb
share/openuniverse/data/stars.dat
@comment MD5:eee6bb0caf1ae32bc2ff043e7baee17a
share/openuniverse/data/messier.dat
@comment MD5:acd357ee82d95121fbf42ba9982f1dd8
```

Конечно, первая строка (❶) – это имя. В следующей строке после ключевого слова `cwd` указано дерево каталогов, в котором был установлен пакет (❷). Можно увидеть, что пакет openuniverse установлен в */usr/X11R6*. Ключевое слово `pkgdep` (❸) предшествует другим пакетам, от которых зависит этот пакет. Комментарий `ORIGIN` (❹) – это категория в дереве «портов», где был создан этот пакет.

Последним идет список файлов (❺). Здесь представлен каждый файл, установленный программой, а также его контрольная сумма MD5. (Различные инструменты, работающие с пакетами, используют контрольную сумму MD5 для верификации файла и подтверждения того, что он не был поврежден при передаче или вследствие ошибки оператора.)

Каждый файл представлен относительно дерева каталогов, указанного в описи. Например, файл *bin/openuniverse* на самом деле установлен в */usr/X11R6*. Таким образом, полный путь к нему такой: */usr/X11R6/bin/openuniverse*. Точно так же различные файлы, представленные как находящиеся в *share/openuniverse*, расположенном в */usr/X11R6*, на самом деле находятся в каталоге */usr/X11R6/share/openuniverse*. Большинство файлов, устанавливаемых в каталоге *share*, относятся либо к документации, либо к данным программ. Можно прочесть документацию или просто запустить openuniverse и посмотреть, что произойдет.

(Значительная часть информации о файлах и каталогах также доступна через pkg_info(1), однако зачастую проще поискать ее самому.)

Деинсталляция пакетов

Для деинсталляции пакетов применяется pkg_delete(1):

```
# pkg_delete openuniverse-1.0.b3
#
```

Если надо деинсталлировать пакет, который необходим другим пакетам, решайтесь на это, только если вы точно знаете, что вы делаете и зачем. (Например, вы хотите обновить пакет зависимостей до более новой версии.) Однако не рассчитывайте, что программа, которой этот пакет необходим, будет выполняться после его деинсталляции!

Для принудительной деинсталляции пакета можно применить `pkg_delete -f`. Программа pkg_delete выдаст предупреждение, но, так или иначе, свою работу выполнит.

Информация о пакете

Деинсталляция работает хорошо, когда вы помните точный номер версии каждого установленного пакета. Если так, примите мои поздравления. Однако если вы, как и я, радуетесь, что запомнили хотя бы названия программ, установленных в системе, то что тут говорить о номерах версий!

В состав FreeBSD входит программа pkg_info – инструмент для изучения установленных пакетов, который более удобен, чем ручное сканирование */var/db/pkg*. В своей работе программа pkg_info(1) опирается на содержимое */var/db/pkg*, однако при этом автоматически выполняет поиск информации и ее сортировку.

При запуске pkg_info без ключей будет выдан список всех пакетов, установленных в системе, а также их краткое описание:

```
# pkg_info
Hermes-1.3.2        Fast pixel formats conversion library
JX-1.5.3_1          A C++ application framework and widget library for X11
Mesa-3.4.2_1        A graphics library similar to SGI's OpenGL
ORBit-0.5.8_1       High-performance CORBA ORB with support for the C language
XFree86-aoutlibs-3.3.3 XFree86 a.out compatibility libraries
...
```

Как видите, в выводе представлены имя и версия каждого установленного пакета, поэтому их можно легко деинсталлировать.

Аргументы pkg_info

Различные аргументы pkg_info позволяют собрать дополнительную информацию о пакетах, установленных в системе. При использовании аргументов необходимо указать либо имя определенного пакета, либо ключ `-a`, что означает «for all packages» (для всех пакетов).

Например, для того чтобы выяснить, каким пакетам в системе необходимы другие пакеты, можно поступить так:

```
# pkg_info -aR
Information for Hermes-1.3.2:

Required by:
windowmaker-0.65.0_1
wmakerconf-2.8.1
Information for JX-1.5.3_1:
Required by:
libjtree-1.1.7_1
libjtoolbar-0.5.4_1
code_crusader-2.1.4_1
...
```

Пространство, необходимое для файлов того или иного пакета, определяется при помощи `pkg_info -s packagename`. (Заметим, что здесь будут учитываться только файлы, установленные пакетом, а файлы, *созданные* при его работе, – это совсем другое дело. В конце концов, вы ведь не считаете текстовые файлы и сообщения электронной почты частью офисного пакета?)

Другой типичный вопрос – к какому пакету относится тот или иной файл. Просматривая */usr/local/bin*, можно наткнуться на нераспознаваемый файл, который никогда не использовался. Почему он здесь? Ключ `-W` команды pkg_info позволяет выполнить «реверсивный поиск» файлов и определить, из каких пакетов они произошли:

```
# pkg_info -W /usr/local/bin/xwe
/usr/local/bin/xwe was installed by package xwpe-1.5.22a
#
```

Управление pkg_add

Для управления инструментами, работающими с пакетами, можно применять переменные окружения.

PKG_TMPDIR

Переменная окружения `PKG_TMPDIR` задает каталог, в который распаковываются временные файлы.

Пакет – это архив tarball с дополнительными инструкциями по установке. Для установки пакета его необходимо разархивировать (untar). Если в стандартных каталогах, которые пытается использовать pkg_add, мало места, то разархивирование не будет завершено, а установка прервется. По умолчанию pkg_add обращается к каталогу, задаваемому переменной окружения `TMPDIR`. Если такая переменная не существует, pkg_add ищет свободное место в таком порядке: */tmp*, */var/tmp* и */usr/tmp*.

С помощью переменной `PKG_TMPDIR` для pkg_add можно задать другой каталог, где есть свободное место:

```
# setenv PKG_TMPDIR /usr/home/mwlucas/garbage
#
```

(Эту строку можно добавить в *.cshrc* – тогда эта переменная будет устанавливаться при каждом вашем входе в систему.)

PACKAGEROOT

Переменная окружения PACKAGEROOT определяет сервер FTP, используемый программой pkg_add для автоматической загрузки пакетов. По умолчанию pkg_add -r пытается скачивать пакеты с *ftp.FreeBSD.org*. Однако зачастую можно добиться большей производительности, если вручную выбрать более близкое и менее нагруженное зеркало.

Присвойте переменной PACKAGEROOT имя конкретного сервера и протокол – в виде URL. Например, для загрузки с *ftp3.FreeBSD.org* введите:

```
# setenv PACKAGEROOT ftp://ftp3.FreeBSD.org
#
```

PACKAGESITE

PACKAGESITE – еще одна хорошо известная переменная окружения, указывающая полный путь к репозитарию пакетов. Она применяется, когда необходимо использовать пакеты определенного «выпуска» или в системе есть локальный репозитарий пакетов. (Установка локального репозитария пакетов рассмотрена в разделе «Сборка пакетов» далее в этой главе.)

Присвоим переменной PACKAGESITE значение в виде абсолютного URL:

```
# setenv PACKAGESITE ftp://ftp4.FreeBSD.org/pub/FreeBSD/releases/
4.4-STABLE/packages/All
#
```

Проблемы с пакетами

Система пакетов выглядит великолепно, не правда ли? Да, отчасти. Но с этой схемой связано несколько проблем, особенно с процессом «портирования» программного обеспечения и его синхронизации.

Подавляющее большинство пакетов производятся сторонними организациями, которые выпускают свои программы по расписанию, совершенно независимому от FreeBSD. Когда они выпускают обновленные версии своих программных продуктов, обновляется и пакет FreeBSD, однако между выпуском оригинального программного пакета и «порта» FreeBSD проходит некоторое время. Популярный «порт» может быть обновлен за считанные часы, а старые версии менее используемых «портов» могут жить дни или недели.

Кроме того, пакеты взаимозависимы; для надлежащей работы многих из них необходимы другие пакеты. Когда команда, поддерживающая «порты» FreeBSD, изменяет пакет, это изменение отзывается во всех зависимых пакетах. Вот почему можно увидеть пакеты с именами, подобными windowmaker-0.65.0_1. Здесь _1 означает, что программа, от которой зависит этот пакет, изменилась, и, следовательно, эта версия WindowMaker незначительно отличается от предыдущей. Инкрементные номера версий также могут свидетельствовать о том, что слегка изменился сам «порт»; например, однажды был об-

новлен процесс сборки для «порта». (Зачастую эти изменения исключительно внутренние и не влияют на работу программы или ее производительность.)

Если вы применяете пакеты, относящиеся только к одному «выпуску» FreeBSD, взаимозависимость не настолько важна. В конце концов, пакеты, собранные для «выпуска», не изменяются. Возможно, вы запускаете предшествующую версию FreeBSD, но хотите выполнять программу, которая только что выпущена. Можно постоянно обновлять операционную систему и при этом иметь старые версии программных продуктов.

Например, для пакета wmakerconf-2.8.1 необходим пакет windowmaker-0.65.0_1. Если установлен подходящий пакет, это замечательно. Однако если установлен windowmaker-0.65.0 или windowmaker-0.65.0_2, программа pkg_add предположит, что требуемый пакет в системе не установлен, и загрузит соответствующий WindowMaker. В лучшем случае это повлечет за собой поглощение дискового пространства, а в худшем будет перезаписана существующая программа.

Один из способов обойти эту проблему заключается в том, чтобы всегда применять пакеты с одной и той же датой или временем. (Если переменная окружения `PACKAGESITE` указывает на каталог пакетов для определенного «выпуска» FreeBSD, пакеты всегда будут согласованы.) Во многих случаях это совершенно приемлемо, поскольку вовсе не обязательно устанавливать последнюю версию той или иной программы, если версия, вышедшая месяц или два назад, работает замечательно. Однако в некоторых случаях такой подход неприменим, поскольку в старой версии могут быть бреши в защите или низкая производительность. В таком случае я рекомендую применять «порты». Вместо проверки установленных программ по имени пакета «порты» проверяют наличие самой программы. Продолжим рассмотрение примера с WindowMaker. «Порт» для wmakerconf не будет искать WindowMaker версии 0.65.0_2, а просто найдет программу, называемую «windowmaker». Такой подход делает «порты» намного более гибкими.

Принудительная установка

Иногда хочется продолжить работу с пакетом, для которого изменилась зависимость, но не хочется обновлять зависимость или применять старый пакет. Не продолжайте. А если вы это сделаете, то не удивляйтесь серьезным сбоям в работе программ.

Тем не менее при желании можно выполнить принудительную установку, – в конце концов, часто изменения в зависимостях незначительны и не влияют на работу программы. Самое трудное – убедиться, что с программами все в порядке.

Примечание

Прежде чем вы продолжите чтение, позвольте предостеречь вас от такого подхода. Оказавшись в такой ситуации, применяйте «порт». Это займет больше времени, но программа наверняка будет работать правильно. Если такой возможности нет, читайте дальше.

Стоит ли проводить принудительную установку? Руководствуйтесь следующим правилом: если имя пакета изменилось за счет добавления символа подчеркивания и номера либо этот номер был инкрементирован, то пакет, возможно, будет работать. Однако никаких гарантий дать нельзя, поэтому при появлении сбоев необходимо деинсталлировать пакет и установить его правильно.

Для проведения принудительной установки сначала загрузите выбранный пакет (убедитесь в том, что компакт-диск с пакетами не монтирован, иначе система будет искать и устанавливать соответствующие зависимости, перезаписывая существующие программы и вызывая неполадки).

Получив пакет, запустите `pkg_add -f`:

```
# pkg_add -f packagename.tgz
```

На экране появится предупреждение о том, что зависимость не найдена, однако процесс установки продолжится. Если программа работает, великолепно! Если нет, деинсталлируйте ее и приступайте к работе с «портами».

Применение «портов»

Сборка программного обеспечения с помощью «портов» займет больше времени по сравнению с применением пакетов. Кроме того, системе «портов» необходимо устойчивое соединение с Интернетом. Тем не менее результаты применения системы «портов» могут оказаться лучше, чем результаты применения пакетов.

Рассмотрим «порт». Обратимся к одному из моих любимых инструментов для обеспечения безопасности, SKIP.[1]

```
# cd /usr/ports/security/skip
# ls
❶ Makefile        ❸ distinfo        ❺ pkg-comment     ❼ pkg-plist
❷ README.html     ❹ files           ❻ pkg-descr       ❽ scripts
#
```

Makefile, показанный в предыдущем листинге (❶), содержит базовые команды для сборки этого порта. Если заглянуть в этот файл, можно сразу заметить, что в нем не так уж и много строк. Makefile для отдельного «порта» в основном содержит лишь базовую информацию о самом «порте», в нем нет общих сведений о том, как собирать программы в FreeBSD. (Большинство

[1] SKIP – это безопасный протокол без установления соединения, разработанный компанией Sun (Secure Connectionless Internet Protocol). Акроним SKIP означает Simple Key-management for Internet Protocols (простой протокол обмена ключами для Интернета, или простое управление ключами для межсетевых протоколов). Это замечательный протокол для виртуальных частных сетей (Virtual Private Networks, VPN), который, к сожалению, вышел из моды с появлением IPSec. Это еще один пример того, как рыночная дубинка может свести в могилу отличную технологию.

Makefile «портов» FreeBSD расположены в каталоге */usr/ports/Mk*; их редактирование – это высший пилотаж, и прежде чем на него решиться, надо хорошо освоить Makefile.)

Файл *README.html* (❷) дает краткое описание «порта». Если для просмотра коллекции «портов» применяется веб-броузер, то по ссылке на информацию об этом «порте» можно перейти как раз к этому файлу.

Файл *distinfo* (❸) содержит информацию для проверки целостности (или контрольные суммы) файлов, необходимых для сборки этой программы.

Каталог *files* (❹) содержит все дополнительные файлы, необходимые для сборки этого «порта». В рассматриваемом примере необходимо 87 заплаток, однако многие «порты» вообще не имеют такого каталога; они собираются «чисто», без заплаток.

Файл *pkg-comment* (❺) содержит описание «порта», представленное одной строкой. Аналогично, *pkg-descr* (❻) содержит более подробное описание и (как правило) URL, по которому можно получить дополнительную информацию о программе.

В файле *pkg-plist* (❼) находится список всех файлов, устанавливаемых «портом» (опись). Если файла в ней нет, он не будет установлен.

Наконец, каталог *scripts* (❽) содержит множество сценариев, выполняемых на различных этапах сборки «портов». Этого каталога может и не быть, – если «порт» собирается без специальной настройки, то у него нет дополнительных сценариев. До и после сборки «порта» эти сценарии выполняют соответственно предварительную и последующую обработку, необходимую «порту», – например, изменение прав доступа на загруженный distfile обеспечит правильную работу программы patch(1).

В совокупности эти файлы образуют инструменты и команды, необходимые для сборки программного обеспечения.

Установка «порта»

Если вы имеете представление об исходном коде, то быстро заметите, что в «порте» нет исходного кода как такового. Конечно, есть заплатки на исходный код и сценарии для его обработки, однако исходного кода нет! Возникает справедливый вопрос: как эта схема работает без исходного кода?

При активизации «порта» система автоматически загружает соответствующий исходный код с проверенного сайта в Интернете. После этого загруженный код проверяется на наличие ошибок целостности и извлекается в рабочий каталог. Далее на него накладываются заплатки, происходит сборка программы, установка всего необходимого и регистрация установки в */var/db/pkg*. Если у порта есть зависимости, которые не установлены, то процесс make прервется на сборку этих зависимостей, а затем возобновится. Чтобы инициировать все это, войдите в каталог «порта» и наберите такую команду:

```
# make install
```

После этого по экрану пробежит множество строк текста, в последней из которых будет содержаться сообщение «recording installation» (запись информации об установке).

Такой целостный процесс установки учитывает все изменения в зависимостях. Если «порту» необходима другая программа, то он просто «закроет глаза» на минимальные изменения в этой программе. Например, пусть несколько месяцев назад в системе установлена версия Apache. Пакет потребовал бы установления новой версии, а «порт» лишь проверит, установлен ли Apache.

Однако по мере приобретения опыта в работе с исходным кодом вы убедитесь, что целостный подход применим не во всех случаях. Беспокоиться не о чем – система «портов» предоставляет возможность организовать процесс сборки «порта» по усмотрению пользователя, поскольку `make install` на самом деле представляет собой серию команд.

Применение make install

Процесс `make install` начинается с `make fetch`. На этом этапе make проверяет наличие исходного кода в */usr/ports/distfiles*. Если его там нет, система постарается его получить.

`make fetch`

Процесс `make fetch` начинает с проверки наличия исходного кода в `MASTER_SITE`, указанном в Makefile, а затем обращается к резервным сайтам, предоставляемым самой системой «портов». Если исходный код найден, он загружается. Этот код называется *distfile*.

`make checksum`

На следующем этапе `make checksum` подтверждает, что цифровая подпись файла *distfile* соответствует подписи, присутствующей в файле *distinfo* «порта». Это мера безопасности – если сервер FTP был взломан злоумышленником, а исходный код заменен на троянского коня либо был поврежден, это обнаружится именно здесь. Процесс сборки будет остановлен с выдачей сообщения о несоответствии контрольной суммы. Если *distfile* был изменен, `make checksum` остановит компиляцию.

Примечание

Иногда разработчики программного обеспечения вносят незначительные изменения в свой код, однако оставляют у файла исходного кода то же имя. После таких изменений «порт» FreeBSD может и не работать. Если вы уверены, что *distfile* не был подменен или поврежден, и хотите использовать его несмотря на предупреждение, замените эту команду на `make NO_CHECKSUM=YES`. Настоятельно рекомендую связаться с поставщиком программы и выяснить, допустимо ли такое изменение.

`make depends`

На этапе `make depends` проверяется зависимость «порта» от любых других программ. Если такие программы есть, то проверяется, установлены ли

они. (Так, для X window manager необходим X server.) Если программа, от которой зависит «порт», не найдена, система рекурсивно обойдет различные зависимости и полностью их соберет.

`make extract`

Получив *distfiles* «порта», их надо развернуть и извлечь. Это происходит в рабочем каталоге «порта». Для создания этого каталога и развертывания *distfiles* применяется `make extract`.

`make patch`.

На этапе `make patch` накладываются все заплатки FreeBSD, перечисленные в Makefile «порта».

`make configure`

Далее `make configure` проверяет необходимость запуска сценария configure. Если сценарий нужен, он выполняется. Если нет, сборка «порта» молча переходит к следующему этапу.

`make build`

На этапе `make build` происходит компиляция проверенного, извлеченного и исправленного программного кода.

`make install`

Наконец, на этапе `make install` устанавливается программа, а ее наличие регистрируется в */var/db/pkg*.

`make target dependencies`

Каждая команда `make target` зависит от предшествующих команд `make target`. Например, нельзя накладывать заплатки на исходный код, который еще не получен. При запуске любой команды `make target` программа make пройдет через все предыдущие этапы, которые еще не были выполнены. Например, команда `make extract` выполнит `make fetch`, `make checksum` и `make extract`.

Как эти этапы применять на практике? Скажем, требуется наложить заплатки на программу до ее компиляции – эти заплатки могут способствовать устойчивости программы и закрывать бреши в защите. Такие заплатки накладываются после извлечения исходного кода и наложения специальных заплаток FreeBSD. Для этого надо запустить `make patch`, наложить новые заплатки к программе в рабочем каталоге согласно инструкциям производителя, а затем вернуться в каталог «порта» и набрать `make install`.

Встроенные функции «портов»

«Порты» позволяют выполнять тонкую подстройку под те или иные требования. Об этом можно прочесть в *Makefile* «порта». Поскольку в этом файле представлены специальные команды для сборки определенной программы, именно здесь вероятнее всего можно найти элементы выбора той или иной опции программы.

При выполнении `make install` многие «порты» сообщают о дополнительных функциях, однако не все из них будут работать (особенно старые). Работоспособность той или иной функции зависит от мастерства разработчика, наличия у него времени и желания заниматься ее поддержкой – помните, что это добровольный проект!

Информацию о дополнительных функциях «порта» всегда можно найти в Makefile, однако сначала рассмотрим это сообщение и научимся его применять. Например, при установке */usr/ports/security/snort* будет выдано уведомление:

```
Set WITH_FLEXRESP, WITH_MYSQL, WITH_ODBC or WITH_POSTGRES
to get additional support.
```

Поиск в Сети позволяет выяснить, что mysql, odbc и postgresql представляют собой пакеты баз данных, а это сообщение уведомляет, что пакет Snort можно собрать с поддержкой этих баз данных. Такой же поиск покажет, что flexresp является частью пакета программ libnet.

Если вы получили уведомление и хотите указать один из этих параметров, нажмите `Control-C`, чтобы прервать сборку «порта». Параметр можно указать в командной строке install:

```
# make WITH_ODBC=YES install
```

Такая команда изменит порядок сборки «порта» – будет собрана версия с поддержкой подключений к базе данных ODBC. Эта функция не встроена в Snort, но вы сможете по сети записывать данные в любую базу данных, которая поддерживает ODBC, например Microsoft SQL Server или Oracle.

В этом «порты» превосходят пакеты. В пакетах такая подстройка невозможна, если нет нескольких версий одного и того же пакета. Упорядочить snort-odbc-1.9.tgz, snort-mysql-1.9.tgz, snort-postgres-mysql-libnet-1.9.tgz и многие другие относящиеся к Snort пакеты было бы крайне трудно. При этом расходовалось бы дополнительное пространство на компакт-диске, а вся затея не нашла бы применения.

Makefile сам сообщает параметры сборки «порта». В начале этого файла можно найти строки, описывающие «порт»:

```
❶ PORTNAME=       snort
❷ PORTVERSION=    1.8
❸ CATEGORIES=     security
❹ MASTER_SITES=   http://www.snort.org/Files/ \
                http://www.physik.TU-Berlin.DE/~ibex/ports/distfiles/
❺ DISTNAME=       ${ PORTNAME} -${ PORTVERSION} -RELEASE
❻ MAINTAINER=     dirk@FreeBSD.org

❼ GNU_CONFIGURE=  yes
❽ CONFIGURE_ARGS= --with-mysql=no --with-odbc=no --with-postgresql=no
❾ MAN8=           snort.8
```

Большая часть этих строк очевидна для администраторов, привыкших собирать программное обеспечение из исходного кода. Если вы пока не находи-

тесь на этой стадии, не стоит волноваться. Со временем вы придете к этому. Рассмотрим данный пример. Не все записи здесь обязательны; в Makefile «порта» может быть много других записей, но приведенные здесь достаточно типичны.

PORTNAME (❶) – это название, обозначающее программы в системе «портов» FreeBSD. Как мы уже видели в примере с Midnight Commander, оно необязательно совпадает с именем самой программы. PORTVERSION (❷) – номер версии программы, который указывается ее разработчиком.

CATEGORIES (❸) перечисляет все каталоги «портов», где можно найти тот или иной порт. Например, этот порт находится в */usr/ports/security*.

MASTER_SITES (❹) содержит список сайтов Интернета, где может быть найдена программа. Именно с них система «портов» пытается загрузить программу. Если один сайт недоступен, она пробует следующий.

DISTNAME (❺) – имя оригинального файла исходного кода программы. Система «портов» старается получить этот файл из MASTER_SITES, указанных ранее.

MAINTAINER (❻) – разработчик, ответственный за поддержку «порта» FreeBSD. На самом деле он не пишет программный код, но отвечает за успешную установку программы в системе FreeBSD.

GNU_CONFIGURE (❼) сообщает системе «портов» о необходимости использовать классическую программу GNU autoconf. В связанной с ней записи, CONFIGURE_ARGS (❽), перечислены аргументы для autoconf.

Обычно есть список страниц руководства, которые устанавливает программа (❾). Прочитав эти страницы с помощью man(1), можно узнать, как использовать эту программу. Далее может идти группа операторов if defined:

```
.if defined ❶ (WITH_FLEXRESP)
BUILD_DEPENDS ❷ += ${ LOCALBASE} /lib/libnet.a:$ ? { PORTSDIR} /net/libnet
CONFIGURE_ARGS +=--enable-flexresp
CONFIGURE_ENV += CPPFLAGS="-I${ LOCALBASE} /include"
LDFLAGS+=       "-L${ LOCALBASE} /lib"
.endif
```

Это опция сборки «порта». Первая строка в данном примере – это переменная (❶), которую надо установить (в нашем случае WITH_FLEXRESP). Вторая строка показывает зависимости (❷), добавляемые к «порту»: */usr/ports/net/libnet*. Остальные строки – это группа команд для сборки программы, которые определяются значением этой переменной.

Переменную WITH_FLEXRESP можно установить из командной строки. Для этого надо набрать:

```
# make install WITH_FLEXRESP=YES
```

Необязательно понимать этот листинг (с if defined) уже сейчас, однако следует обратить внимание на маленький знак вопроса (?) и символы «плюс» или «равно» (+=), рассеянные по строкам. Они означают следующее: чтобы добавить те или иные команды в процесс сборки программы, достаточно

установить значение соответствующей переменной! Установить значение переменных в командной строке намного проще, чем выяснять, как самому добавлять команды в этот процесс.

Если при сборке «порта» применяется несколько команд make (например, `make patch` и затем `make install clean`), необходимые опции нужно включать в каждую команду. В противном случае «порт» может быть собран неправильно. Например, если для Snort нужна опция flexresp, но вы хотите наложить свою собственную заплатку, необходимо выполнить следующее:

```
# make patch WITH_FLEXRESP=YES
```

Затем надо наложить свою заплатку и выполнить такую команду:

```
# make install WITH_FLEXRESP=YES
```

В противном случае на систему будет наложена заплатка с параметром синхронизации `WITH_FLEXRESP`, но система не получит правильных инструкций при компиляции программы. Программа будет внутренне противоречива и вполне может привести к сбоям.

Самая сложная часть в сборке программы – решить, какие опции применять. К сожалению, на этот вопрос нет простого ответа, поэтому для принятия решения лучше всего изучить руководство по этой программе или зайти на соответствующий веб-сайт. Мне неоднократно приходилось устанавливать программы, читать документацию и возвращаться обратно, чтобы деинсталлировать программу и установить ее заново с учетом необходимых параметров.

Деинсталляция и повторная установка

Для установки «портов» характерна замечательная особенность: после того как «порт» установлен, он воспринимается как пакет. Применив pkg_delete, его можно деинсталлировать, а с помощью pkg_info – получить информацию о нем. Так как установка «портов» регистрируется в */var/db/pkg*, можно изучить файл *contents* и узнать, какие файлы входят в состав «порта».

Кроме того, «порт» можно деинсталлировать в каталоге «порта». Например, пусть в состав FreeBSD входит несколько версий одного «порта», скажем, веб-сервера Apache. Можно собрать различные версии, оценить их и выбрать одну для установки на длительный срок. Для деинсталляции версии, прошедшей проверку, можно запустить `make deinstall` в каталоге «порта». При этом программа будет удалена из основной системы.

Примечание

После выполнения `make install` скомпилированная программа и файлы исходного кода по-прежнему расположены в рабочем подкаталоге «порта». Для повторной установки деинсталлированной программы можно запустить `make reinstall`. Деинсталлировать и повторно устанавливать программу можно сколько угодно раз.

В какой-то момент может потребоваться повторно установить «порт», удаленный с помощью pkg_delete. Однако команда `make reinstall` не выполнится;

вместо этого будет выдано сообщение о том, что «порт» уже установлен. Почему? Рассмотрим внимательнее рабочий каталог «порта»:

```
# cd /usr/ports/www/apache13/work
# ls -a
total 21
969094 drwxr-xr-x  3 root  wheel    512 Jul 22 21:24 .
778196 drwxr-xr-x  4 root  wheel    512 Jul 27 20:42 ..
969343 -rw-r--r--  1 root  wheel  17163 Jul 22 21:24 .PLIST.mktmp
969344 -rw-r--r--  1 root  wheel      0 Jul 22 21:24 .PLIST.setuid
969345 -rw-r--r--  1 root  wheel     19 Jul 22 21:24 .PLIST.startup
969341 -rw-r--r--  1 root  wheel      0 Jul 22 21:23 .build_done
969115 -rw-r--r--  1 root  wheel      0 Jul 22 21:22 .configure_done
969112 -rw-r--r--  1 root  wheel      0 Jul 22 21:21 .extract_done
969342 -rw-r--r--  1 root  wheel      0 Jul 22 21:24 .install_done
969113 -rw-r--r--  1 root  wheel      0 Jul 22 21:21 .patch_done
318289 drwxr-xr-x  8 root  wheel    512 Jul 22 21:23 apache_1.3.20
#
```

Итак, о чем говорит этот листинг? Все файлы, начинающиеся с точки (в данном случае большинство файлов, за исключением *apache_1.3.20*), являются «скрытыми» файлами, которые не отображаются при обычном просмотре каталога. Система «портов» и процесс make используют эти файлы для отслеживания этапов процесса сборки. Эти файлы применяет каждый порт. Видите скрытый файл *.install_done*? Если этот файл существует, «порт» полагает, что все уже установлено, и отказывается перезаписывать самого себя. В этом и заключается проблема.

Удалите этот файл, и команда `make reinstall` будет успешно выполнена.

Очистка с помощью make clean

«Порты» могут занимать много места. Некоторые «порты» могут поглотить пару сотен мегабайт диска в ходе извлечения исходного кода и сборки программы. Бульшую часть этого пространства занимает именно исходный код, в котором больше нет необходимости. В системе «портов» есть метод, позволяющий удалить избыточный исходный код.

Когда программа установлена и сконфигурирована, исходный код в каталоге «порта» больше не нужен. Его можно удалить с помощью `make clean`. (Эта операция «снесет» рабочий каталог, поэтому перед ее выполнением необходимо убедиться, что с установленной программой все в порядке!) Кроме того, очистить новый «порт» можно сразу же после установки. Для этого установку надо выполнять так: `make install clean`.

Также можно очистить исходные *distfiles* «порта», которые хранятся в */usr/ports/distfiles*. (Время от времени проверяйте этот каталог: если устанавливается много «портов», он заполнится очень быстро.) Удаление ненужных *distfiles* высвободит значительный объем дискового пространства.

Для очистки всего дерева «портов» запустите `make clean` прямо в */usr/ports*. Этот процесс займет какое-то время. Хотя есть более быстрые и эффектив-

ные способы удаления каждого рабочего каталога в дереве «портов», описанный метод напрямую поддерживается Проектом FreeBSD.

Сборка пакетов

Если вы используете «порты», то можете собирать свои пакеты и устанавливать их на других машинах FreeBSD, что поможет сберечь время и обеспечит идентичность программного обеспечения на каждой машине. Например, если программа Snort, применяемая на нескольких машинах, должна работать одинаково на каждой из них, можно собрать Snort один раз и затем установить этот пакет на всех других машинах.

Команда для создания пакета такова: `make package`. Она установит программу на локальной машине и создаст пакет в каталоге «порта». Далее надо скопировать этот пакет на другие системы и установить их с помощью pkg_add.

Можно даже установить анонимный сервер FTP server (см. главу 12) и организовать локальный master-репозитарий пакетов. Помните переменную окружения `PACKAGESITE`? Установите ее согласно пути к анонимному серверу FTP и разместите на нем свои пакеты. Далее на других машинах можно выполнять `pkg_add -r`, что позволит автоматически забирать и устанавливать самодельные пакеты.

Изменение пути установки

Если у вас есть десятки и даже сотни систем FreeBSD с идентичной конфигурацией, то путь установки */usr/local*, устанавливаемый по умолчанию для «портов» и пакетов, может вызывать затруднения. Во многих компаниях с большим количеством серверов каталог */usr/local* зарезервирован для программ, уникальных для отдельной машины, а другие программные пакеты, используемые в каждой системе, должны устанавливаться в любом другом месте.

Обычная альтернатива для */usr/local* – каталог */usr/pkg*, который можно задать с помощью переменной `PREFIX`:

```
# make PREFIX=/usr/pkg install clean
```

Устанавливаемые «порты» будут размещаться в выбранном каталоге.

Установка параметров make по умолчанию

Если вы устали многократно набирать одни и те же параметры при сборке «портов», можете перечислить их в *make.conf*, и тогда они будут автоматически применяться при установке каждого «порта». Более подробную информацию можно найти в разделе главы 9, посвященном *make.conf*.

Примечание

Кстати, файл */etc/make.conf* сканируется при каждом запуске make. Это означает, что все параметры, установленные в нем, будут применяться к «портам». Параметры, подобные `CPUTYPE`, могут показаться незначительными, но иногда приобретают важное значение. Они могут стать причиной путаницы. Поэтому необходимо хотя бы знать об их существовании.

Обновление «портов» и пакетов

Процесс обновления программного обеспечения может быть очень прост или стать настоящим кошмаром. Избежать большей части подводных камней поможет предварительная подготовка. Приводимый далее список рекомендаций предполагает, что вы обновляете сервер Интернета, от которого зависят реальные пользователи. (Если вы обновляете программы на своем лэптопе, можете считать, что вы предупредили своего пользователя.)

1. При обновлении сначала необходимо убедиться в наличии предшествующей версии устанавливаемого пакета. Если возникнут сбои на рабочей системе, можно будет быстро вернуться к предыдущей версии пакета. Если вы устанавливали программу с компакт-диска, убедитесь, что он рядом, а если установка проходила через FTP, скачайте этот пакет и держите его под рукой.
2. По возможности тестируйте обновленное программное обеспечение на тестовой машине. Обновления на рабочем сервере могут добавить седых волос и испортить настроение даже опытному администратору. Обновление, проведенное удачно, позволит легче и быстрее провести обновление на другой машине.
3. Убедитесь в наличии резервной копии системы. В главе 3 подробно рассмотрено резервное копирование с применением лент или файловой системы.
4. Подготовьте обновленное программное обеспечение. Лучше, если это будет пакет, собранный на тестовой машине (теперь вы знаете, что «порт» на самом деле собирается и устанавливается). В противном случае соберите программу из «порта», запустив `make build`. Не выполняйте `make install` — выполните только `make build`, чтобы убедиться в успешной компиляции программы.
5. Известите пользователей о том, что тот или иной сервис будет обновляться в такое-то время, и в этот период программа или машина будут недоступны.
6. В запланированное время выполните `make deinstall` или `pkg_delete` для предшествующей версии пакета, а затем с помощью `make install` соберите новый «порт». Будьте готовы вернуться к предшествующей версии, если новая не будет работать!

При обновлении чаще всего трудно определить, какие программы в нем нуждаются. Я следую такому правилу: работающую программу не надо обновлять только потому, что вышла новая версия. Это правило особенно подходит для больших, сложных программных пакетов, таких как некоторые новые менеджеры рабочего стола.

Тем не менее вы можете посчитать, что хотя программа работает прекрасно, новая версия поможет решить ту или иную проблему или предоставит необходимую функциональность. Облегчить свою работу можно с помощью обновления дерева «портов». Это позволит легко устанавливать новые программы.

Обновление коллекции «портов»

В процессе обновления FreeBSD также происходит обновление дерева «портов». CVSup и конфигурацию в *ports-supfile* можно использовать для обновления портов до последней версии или, по сути, до любой выбранной версии.

Для начала надо установить CVSup, как описано в главе 6, и отредактировать *ports-supfile*, указав конкретное зеркало CVSup. После этого выполните такую команду:

```
# cvsup ports-supfile
```

CVSup медленно проползет по дереву «портов», сравнивая каждый файл с его версией на соответствующем зеркале CVSup, и внесет необходимые изменения в файлы. После выполнения этой команды в системе будет стоять последняя версия дерева «портов».

После обновления коллекции «портов» необходимо обновить индекс и файлы *readme*. Для этого перейдите в */usr/ports* и наберите такую строку:

```
# make index && make readmes
```

Примечание

Каждая из команд (`make index` и `make readmes`) выполняется довольно долго. Применив `&&`, вы связываете их вместе; после успешного выполнения первой команды будет запущена вторая. Такой подход избавит от необходимости через полчаса возвращаться обратно и набирать вторую команду.

При обновлении коллекции «портов» старые рабочие каталоги не удаляются. Если вы установили «порты» и не выполнили `make clean`, рабочие каталоги и старый исходный код, включая скрытые файлы, по-прежнему останутся на диске. Если запустить make в этих «портах», скрытые файлы заявят, что порт уже установлен, а команда make не будет выполнена. Для сборки таких «портов» необходимо предварительно запустить `make clean`.

Вопросы обновления коллекции «портов»

Сложность обновления коллекции «портов» проявляется в том, что любая коллекция программного обеспечения предназначена для использования как одно целое. Представим, что у вас есть старая версия распространенного инструмента, который необходим для новой программы. Возможно, этот инструмент, а также зависящие от него программы придется обновить. Опасность заключается в том, что простое обновление может вызвать каскад последующих обновлений.

Конечно, FreeBSD – не единственная операционная система, испытывающая такие затруднения. Они характерны для любого программного пакета в любой операционной системе. (В Windows это зачастую проявляется в виде конфликтов DLL, необъяснимых сбоев программ и любых других странных и неприятных признаков.) Отличная схема программы минимизирует, но не устраняет эту загвоздку. К сожалению, отличные схемы программ редки.

Проверка версий программ

На специализированной машине последовательное обновление не настолько сложно; в конце концов, на веб-сервере обычно не устанавливаются сотни программных пакетов. Однако на рабочей станции такое бывает. Даже в моем лэптопе более 200 записей в */var/db/pkg*! (А ведь мне и в самом деле надо пройтись по ним и деинсталлировать ненужные пакеты, действительно ли мне необходим этот чертик, преследующий мою мышь?) Итак, что предпринять, если в системе есть сложные программные зависимости?

В состав FreeBSD входит инструмент pkg_version(1), предназначенный для проверки версий программного обеспечения. pkg_version сравнивает версию установленной программы с версией в */usr/ports/INDEX*. Если файл *INDEX* обновляется своевременно, то все в порядке. (Следуете ли вы моему совету из последнего раздела? Вы обновляете индексы и файлы *readme*, верно? Конечно, да. Ведь вы не из тех, кто тратит с трудом заработанные деньги на компьютерную книгу, а потом пренебрегает ее советами? Конечно, нет.)

Результат элементарной проверки версий может выглядеть так:

```
# pkg_version -v
apache-1.3.20                    =   up-to-date with port
autoconf-2.13_1                  =   up-to-date with port
bzip2-1.0.1                      =   up-to-date with port
cvsup-bin-16.1                   ?   orphaned: net/cvsup-bin
emacs-20.7                       =   up-to-date with port
gettext-0.10.35                  =   up-to-date with port
gmake-3.79.1                     =   up-to-date with port
ispell-3.1.20c_2                 <   needs updating (port has 3.2.04_1)
jade-1.2.1_1                     =   up-to-date with port
libtool-1.3.4_2                  =   up-to-date with port
links-0.96pre7                   <   needs updating (port has 0.96,1)
m4-1.4                           =   up-to-date with port
mutt-1.2.5                       =   up-to-date with port
rrdtool-1.0.33                   =   up-to-date with port
sftp-0.9.6_1                     =   up-to-date with port
sudo-1.6.3.7                     <   needs updating (port has 1.6.3.7_1)
ucd-snmp-4.2.1                   =   up-to-date with port
unzip-5.42                       =   up-to-date with port
uulib-0.5.13                     =   up-to-date with port
xsysinfo-1.4a                    =   up-to-date with port
zip-2.3                          =   up-to-date with port
```

Прочитав комментарии для каждой программы, можно легко заметить, что большинство программ представлены последними версиями. Однако обратите внимание на запись для ispell-1.2.10. Сообщение говорит, что этот «порт» устарел. Когда-нибудь вы захотите обновить эту программу. Надо решить, настолько ли важна эта программа, чтобы тратить время на ее обновление.

Теперь, поскольку я сам установил каждую программу на этой системе, я хорошо в ней ориентируюсь. Я знаю, насколько важна та или иная программа. Я знаю, что ispell – это программа для проверки орфографии и что для веб-сервера ее важность минимальна (в лучшем случае). И я не буду о ней беспокоиться. С другой стороны, пакет sudo-1.6.3.7 – это инструмент для обеспечения безопасности, который применяется для управления правами доступа пользователей, и правильная работа этой программы чрезвычайно важна. Если доступна новая версия, я должен ее изучить и, возможно, обновить программу.

Запись для cvsup-bin-16.1 с сообщением «orphaned: net/cvsup-bin» говорит о том, что для этой программы нет записи в */usr/ports/INDEX* и, следовательно, нет и «порта» для этого пакета. Я инсталлировал этот «порт» из пакета; «порта» как такового нет в наличии.

Автоматическая проверка версий программ

Автоматическую проверку версии программ можно добавить к еженедельным операциям, уведомление о которых приходит по электронной почте. Для этого добавьте в */etc/periodic.conf* следующую строку:

```
weekly_status_pkg_enable="NO"
```

Создайте файл */etc/periodic.conf*, если его нет в системе. Его полное описание можно найти в главе 9.

Советы по обновлению

Большая часть операций по обслуживанию программного обеспечения определяется предназначением того или иного сервера. Если вы – единственный администратор машины, то все очень просто. Однако если администраторов несколько, отслеживать всю информацию становится очень нелегко. Трудно переоценить важность ведения протоколов на каждой системе в локальной сети. Даже простой текстовый файл, в который быстренько заносятся записи вида «mwlucas, 5-15-01: установил sftp для клиента *bufar@absolute-bsd.com*», может избавить от многочасовых разбирательств. Тогда будет легче выяснить, почему тривиальные изменения влекут за собой выход системы из строя. Другому администратору не придется звонить вам в 3 часа утра и спрашивать, почему система не работает с настройками по умолчанию.

Решив обновить рабочую систему, спланируйте выполнение изменений. Команда `pkg_info -aR` поможет выяснить, какие пакеты необходимы для того или иного пакета.

При обновлении следует придерживаться правила: начинать с зависимостей. Эти пакеты необходимы другим пакетам, а значит, их надо обновить и протестировать в первую очередь. В конце концов, если что-то не в порядке с программами более низкого уровня, будут сбои и у всех зависимых от них программ.

Для удаления зависимостей можно применять `pkg_delete -f`, а затем установить новые версии из «портов» или пакетов. Далее следует идти по цепочке и обновлять требуемые программы. Повторюсь: можно попробовать запустить программный пакет с новой версией зависимости, но он может и не заработать.

Стоит задуматься о применении инструмента */usr/ports/sysutils/portupgrade*, поскольку он позволяет автоматизировать часть этих задач. Тем не менее вы должны понимать эти операции и уметь справляться с конфликтами и зависимостями вручную.

По договору между издательством «Символ-Плюс» и Интернет-мага-зином «Books.Ru-Книги России» единственный легальный способ получения данного файла с книгой «FreeBSD. Подробное руководство» (ISBN 5-93286-066-9) – покупка в Интернет-магазине «Books.Ru-Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (www.symbol.ru), где именно Вы

11

Расширенное управление программным обеспечением

Эта глава охватывает аспекты, понимание которых желательно при запуске программного обеспечения во FreeBSD.

FreeBSD поддерживает работу множества различных программных продуктов. Большая их часть доступна в виде исходного кода, поэтому их можно собрать как «родное» программное обеспечение FreeBSD. И благодаря продуманной организации FreeBSD в ней также можно запускать программы из многих других операционных систем. Посмотрим, как это можно реализовать. Внимание будет сосредоточено на популярном пакете совместимости с Linux, который позволяет запускать во FreeBSD программы Linux без их модификации.

Кроме того, для запуска программ во время начальной загрузки и их останове при выключении системы необходимо редактировать и создавать заново сценарии запуска и останова. Хотя одни программы прекрасно прекращают работу при останове операционной системы, другим программам (например, базам данных) необходима более мягкая остановка работы. Хотя на выручку могут прийти различные трюки, «чистый» запуск и останов сетевых сервисов – это замечательный навык, который нужно приобрести и применять. Здесь будет рассмотрено, как реализовать такие системы и управлять ими во FreeBSD.

И еще. Хотя в обычных условиях необязательно знать, как во FreeBSD организованы связывание и поддержка совместно используемых библиотек, здесь будут обсуждаться вопросы их конфигурирования и управления ими. Почему? Потому что обычные условия, как ни странно, довольно редки в компьютерной среде.

Наконец, здесь будет рассмотрена работа систем с несколькими процессорами и их взаимодействие с программным обеспечением. Хотя несколько процессоров могут значительно усилить мощь системы, они не принесут пользы, если программы не сконфигурированы должным образом.

Сценарии запуска и останова

Основные программы FreeBSD запускаются посредством /etc/rc, но дополнительные программы запускаются отдельными сценариями, которые устанавливаются системой «портов» и пакетов. Однако при установке собственной программы необходимо создать сценарий, который управляет ее запуском и остановом. Для изменения процесса запуска существующего пакета необходимо понимать, как работают сценарии запуска и останова.

Примечание

Этот раздел предполагает базовое понимание командных сценариев. Если вы никогда их не видели и не применяли, изучите примеры очень внимательно. Создавать командные сценарии нетрудно, и лучший способ научиться этому состоит в том, чтобы изучать примеры.

Командные сценарии /etc/rc (см. главу 9) управляют процессом запуска основной системы. В ходе начальной загрузки сценарии запуска FreeBSD проверяют несколько каталогов и запускают найденные в них дополнительные командные сценарии. Чаще всего сценарии запуска и останова находятся в каталоге */usr/local/etc/rc.d*, но могут располагаться по умолчанию и в*/usr/X11R6/etc/rc.d*. Эти каталоги задаются в */etc/defaults/rc.conf* и могут быть подменены в */etc/rc.conf*. (Дополнительные каталоги сценариев можно добавить с помощью переменной `local_startup` в *rc.conf*.)

В процессе поиска командных сценариев система проверит эти каталоги на наличие файлов, имена которых заканчиваются на «.sh». Если система находит такой файл, он расценивается как командный сценарий и выполняется с аргументом `start`. При останове система FreeBSD запускает те же сценарии с аргументом `stop`. Предполагается, что сценарии принимают эти аргументы и выполняют соответствующие действия.

Типичный сценарий запуска

Рассмотрим типичный сценарий запуска, snmpd.sh. Он является частью пакета net-snmp, который будет рассмотрен чуть позже. Сейчас достаточно знать, что этот сценарий запускает сервер SNMP при начальной загрузке и останавливает этот демон при выключении системы. Вот этот сценарий:

```
❶ #!/bin/sh

❷ if ! PREFIX=$(expr $0 : "\(/.*\)/etc/rc\.d/$(basename $0)\$"); then
    echo "$0: Cannot determine the PREFIX" >&2
    exit 1
fi

❸ case "$1" in
start)
       [ -x ${ PREFIX} /sbin/snmpd ] && ${ PREFIX} /sbin/snmpd && echo -n ' snmpd'
       ;;
stop)
       killall snmpd && echo -n ' snmpd'
       ;;
```

```
*)
        echo "Usage: `basename $0` { start|stop} " >&2
        ;;
esac

exit 0
```

Строка #!/bin/sh (❶) показывает, что это командный сценарий. Остальная часть файла (подобно командному файлу в Windows) содержит команды, которые выполняются сценарием.

Первый раздел, ограниченный if (❷) и fi, определяет путь к программам и сообщает следующим командам сценария, в каком каталоге он запущен – в */usr/local*, */usr/X11R6* или в каком-то другом. Оставшейся части сценария (❸) надо знать об этом, чтобы находить соответствующие команды.

В строке case "$1" (❸) сценарий принимает решение о дальнейших операциях. Здесь считывается первый аргумент, с которым был вызван сценарий. Например, если сценарий запущен как snmpd.sh start, то start – это первый аргумент. Если запустить сценарий как snmpd.sh stop, первым аргументом будет stop. В сценарии есть несколько подразделов. Все строки между start) и символами «;;» – это шаги, выполняемые при условии, что первым аргументом является start. Все строки между stop) и следующими двумя символами точка с запятой (;;) – это действия, выполняемые при условии, что первым аргументом является stop. Последний раздел, *), – групповой символ (wildcard) для всех других возможных аргументов.

Например, если запустить этот сценарий как snmpd.sh start, он выполнит следующую команду:

```
[ -x ${PREFIX}/sbin/snmpd ] && ${PREFIX}/sbin/snmpd && echo -n ' snmpd'
```

Это стандартные команды оболочки. Сначала проверяется существование программы snmpd. Если такая программа есть, она запускается и выводит свое имя. Аналогично, если вызвать сценарий с аргументом stop, он запросто уничтожит все процессы snmpd.

Чтобы изменить поведение пакета при запуске, необходимо отредактировать сценарий запуска. Например, для запуска snmpd с аргументом -D надо отредактировать стартовую строку. Найдите часть команды, в которой запускается snmpd, и измените ее:

```
[ -x ${PREFIX}/sbin/snmpd ] && ${PREFIX}/sbin/snmpd -D && echo -n ' snmpd'
```

Применение сценариев для управления запущенными программами

Эти сценарии можно применять не только при запуске и останове, но и в ходе работы системы. Например, чтобы перезапустить демон snmpd и заставить его перечитать свой конфигурационный файл, можно выполнить такую команду:

```
# /usr/local/etc/rc.d/snmpd.sh stop && /usr/local/etc/rc.d/snmpd.sh start
```

Примечание

Для управления запущенными программами вовсе не обязательно применять сценарии. Если вы изучили командный сценарий snmpd.sh, то знаете, что для останова программы сценарий выполняет команду `killall snmpd`, а для старта программы запускается `snmpd`. Эти команды можно ввести прямо в командной строке, и они окажут тот же эффект, что и запуск самого сценария. Есть два варианта: либо помнить о том, какие действия выполняет сценарий для управления той или иной программой, либо набирать полный путь к сценариям запуска. При наличии сомнений используйте сценарии!

Управление совместно используемыми библиотеками

Идея, лежащая в основе совместно используемых библиотек, довольно проста: это фрагмент компилированного кода, предоставляющий сервисы и функции другим фрагментам компилированного кода. Разделяемые библиотеки предоставляют часто вызываемые функции другим программам и спроектированы так, чтобы как можно больше программ могли многократно их использовать.

Например, многие программы должны хешировать данные, или шифровать без возможности расшифровки[1] (one-way encrypt). Однако если та или иная программа включает в себя модули для хеширования, она будет больше, писать ее станет труднее, а поддерживать – менее приятно. Более того, у программ могут возникнуть трудности с взаимодействием, если хеширование в них реализовано по-разному. Если же программа, которой необходимо хеширование, может обратиться к соответствующей функции совместно используемой библиотеки (в данном случае libcrypt), проблемы с взаимодействием и поддержкой устраняются. Точно так же другие совместно используемые библиотеки предоставляют свои функции для поддержки других программ. Такой подход снижает средний размер программ, высвобождая значительный объем системной памяти.

Во время загрузки система FreeBSD создает кэш доступных совместно используемых библиотек. Программы не должны сканировать весь диск при их поиске; они просто запрашивают в кэше необходимые функции. На самом деле способность управлять кэшем библиотек – это одна из черт, отличающих новичка от профессионала.

FreeBSD предоставляет довольно много разумных параметров, задаваемых по умолчанию, но здесь мы рассмотрим инструменты, необходимые для конфигурирования и управления кэшем в различных обстоятельствах. Совме-

[1] Например, для сохранения паролей. Как известно, пароли в современных системах хранятся в зашифрованном виде, а вводимый пользователем пароль шифруется тем же алгоритмом, что и оригинальный пароль. Затем зашифрованный оригинал и зашифрованный введенный пароль сравниваются. – *Примеч. науч. ред.*

стно используемые библиотеки – это средства с норовом, однако с помощью ldconfig, ldd и головы на плечах их можно приручить.

Ldconfig

Основной инструмент для управления совместно используемыми библиотеками – ldconfig(8). (Ссылки на него встречаются в разных местах.) Здесь мы обсуждим несколько различных команд, относящихся к ldconfig: rtld(1), ldd(1) и саму программу ldconfig.

Типы исполняемых файлов: aout и ELF

Прежде всего рассмотрим типы исполняемых файлов: *aout* и *ELF*. Как администратор вы не должны хорошо разбираться в aout и ELF, но должны знать, что aout – это устаревший стандарт, а ELF – новый, и что скомпилированные программы одного типа не могут использовать библиотеки другого типа. ELF-программы не могут применять aout-библиотеки. (Во FreeBSD исполняемый файл Netscape имеет формат aout, поэтому для его применения необходимо установить библиотеки совместимости с aout среды XFree86.) Хотя исполняемые aout-файлы постепенно исчезают, системе FreeBSD еще неопределенно долго придется поддерживать оба типа.

При запуске ELF-программы, которой необходимы библиотеки, система вызывает rtld(1), «компоновщик на этапе выполнения» (run-time linker). rtld исследует загруженные программы, определяет, какие библиотеки им необходимы, а затем загружает эти библиотеки. Для aout-программ есть отдельный компоновщик, называемый ld(aout).

Вместо того чтобы искать библиотеки по всей системе при каждом вызове той или иной программы, rtld берет совместно используемые библиотеки из кэша библиотек. Кэш находится в двух отдельных файлах системы: */var/run/ld.so.hints* (aout) и */var/run/ld-elf.so.hints* (ELF). Неверно сконфигурированный кэш – наиболее вероятная причина проблем с библиотеками.

Какие библиотеки есть в системе?

Для получения списка библиотек, установленных в системе, запустите ldconfig с флагом `-r`:

```
# ldconfig -r
/var/run/ld-elf.so.hints:
        search directories: /usr/lib:/usr/lib/compat:/usr/X11R6/lib:/usr/local/lib:/
usr/local/lib/mysql:/usr/local/pilot/lib
         0:-lcom_err.2 => /usr/lib/libcom_err.so.2
         1:-lscrypt.2 => /usr/lib/libscrypt.so.2
         2:-lcrypt.2 => /usr/lib/libcrypt.so.2
...
```

`ldconfig -r` изучает кэш библиотек и перечисляет все найденные библиотеки. В моей системе список состоит из 229 пунктов.

Примечание

Если программа сообщает, что не может найти библиотеку, изучите вывод `ldconfig -r`. Если требуемой библиотеки в нем нет, то возможны два варианта: либо кэш неверно сконфигурирован или неполон, либо этой библиотеки действительно нет в системе.

Создание кэша

С помощью ldconfig кэш создается в процессе загрузки системы. Для ELF-программ это выглядит примерно так:

```
# ldconfig -elf /list/of /path/names/here
```

Аналогично для aout применяется такая команда:

```
# ldconfig -aout /other/list /of/paths
```

Примечание

Список имен путей задается в `/etc/rc.conf` как `ldconfig_paths` и `ldconfig_paths_aout`.

Если программа обращается к библиотеке, которая только что установлена и отсутствует в кэше, произойдет ошибка. В этом случае необходимо заново создать кэш. Сделать это довольно легко. Просто запустите ldconfig без аргументов; ldconfig повторно просмотрит каталоги, указанные в `ldconfig -r`, и заново создаст кэш.

Поиск библиотеки

Если искомой библиотеки нет ни в одном из просканированных каталогов, ее надо найти. Вообще говоря, если отсечь первые символы «lib» в имени библиотеки и применить `locate` или `find / -name libname -print`, этот файл будет найден. В худшем случае придется пробираться сквозь длинный список строк результата, в одной из которых отыщется нужная библиотека.

Добавление библиотек

После установки той или иной программы в системе может появиться новый каталог с библиотеками. (Иногда их можно найти в собственных подкаталогах программы; например, программа для поддержки PalmPilot на моей машине использует */usr/local/pilot/lib*.)

С помощью ключа `-m` можно достаточно просто «влить» библиотеки нового каталога в существующий кэш. Некоторые «порты» даже применяют этот ключ для конфигурирования библиотек во время начальной загрузки, что исключает утомительную путаницу с */etc/rc.conf*. Для «вливания» моей библиотеки в существующий кэш я набираю такую команду:

```
# ldconfig -m /usr/local/pilot/lib
```

LD_LIBRARY_PATH

Ключ -m замечательно работает с точки зрения системного администратора, однако он не подходит для скромного пользователя без привилегий root.[1] Кроме того, если у обычного пользователя есть свой набор библиотек, системный администратор может не разрешить к ним всеобщий доступ. А поскольку владельцем каталога совместно используемых библиотек должен быть root, то обычные пользователи не смогут получить доступ к этому каталогу. Вероятно, системный администратор постарается не оставить ни малейшего шанса для связывания системных программ с личными библиотеками пользователя.

Именно здесь на помощь придет переменная окружения LD_LIBRARY_PATH. Вместо того чтобы создавать кэш, переменная LD_LIBRARY_PATH предписывает системе искать новые библиотеки в каталогах, список которых она задает.

Примечание

Этот подход небезопасен, – если в LD_LIBRARY_PATH будет задано чрезмерно доступное месторасположение библиотек, ваша программа может быть связана с чем угодно. Кроме того, LD_LIBRARY_PATH подменяет кэш, поэтому будьте внимательны, добавляя в нее что-нибудь!

В LD_LIBRARY_PATH можно указать любое количество каталогов, разделив их двоеточиями. Например, я могу поместить каталоги */home/mwlucas/lib* и */compat/linux/usr/lib/local* в LD_LIBRARY_PATH, чтобы выполнить установку программного обеспечения. Для этого я наберу такую команду:

```
# setenv LD_LIBRARY_PATH /home/mwlucas/lib:/compat/linux/usr/lib/local
```

Например, обычно я устанавливаю StarOffice вручную, а не применяю «порт». При этом процедура установки извлекает различные библиотеки в */tmp/sv001.tmp* и рассчитывает найти их при запуске графического инсталлятора. Чтобы эти библиотеки действительно были найдены, я начинаю установку пакета с того, что задаю в LD_LIBRARY_PATH каталог */tmp/sv001.tmp*:

```
# setenv LD_LIBRARY_PATH /tmp/sv001.tmp
# ./setup
```

Когда будет запущен графический инсталлятор StarOffice, он найдет в этом каталоге извлеченные библиотеки. Таким образом, не надо реконфигурировать всю систему FreeBSD только для того, чтобы применять ту или иную программу.

Примечание

Помните, переменные окружения можно автоматически устанавливать при входе в систему. Для этого их надо задать в файле *.cshrc* или *.profile*.

[1] Если вы читаете эту книгу, то вы, вероятно, и есть системный администратор. Однако это решение необходимо для ваших пользователей.

Какие библиотеки нужны моей программе?

Наконец, возникает вопрос: какие библиотеки нужны программе? Эту информацию можно получить с помощью ldd(1). Например, узнать, что требуется программе Emacs, можно посредством такой команды:

```
# ldd /usr/local/bin/emacs
/usr/local/bin/emacs:
        libXaw.so.6 => /usr/X11R6/lib/libXaw.so.6 (0x28159000)
        libXmu.so.6 => /usr/X11R6/lib/libXmu.so.6 (0x2818e000)
        libXt.so.6 => /usr/X11R6/lib/libXt.so.6 (0x2819f000)
        libSM.so.6 => /usr/X11R6/lib/libSM.so.6 (0x281e2000)
        libICE.so.6 => /usr/X11R6/lib/libICE.so.6 (0x281ea000)
        libXext.so.6 => /usr/X11R6/lib/libXext.so.6 (0x281fe000)
        libX11.so.6 => /usr/X11R6/lib/libX11.so.6 (0x28209000)
        libutil.so.3 => /usr/lib/libutil.so.3 (0x282a2000)
        libm.so.2 => /usr/lib/libm.so.2 (0x282ab000)
        libc.so.4 => /usr/lib/libc.so.4 (0x282c6000)
        libXThrStub.so.6 => /usr/X11R6/lib/libXThrStub.so.6 (0x28361000)
#
```

Этот вывод содержит имена совместно используемых библиотек, которые нужны Emacs, и указывает расположение файлов, содержащих эти библиотеки. Этот список можно сравнить с выводом команды `ldconfig -r`, дабы убедиться, что у программы есть все необходимое. Либо этот список можно использовать как «список необходимых покупок» и поискать необходимые библиотеки.

Запуск программного обеспечения из чужой ОС

В операционных системах применяется программное обеспечение, написанное специально для них, и программы могут выполняться только на той платформе, для которой они разработаны. Как известно, многие специалисты сделали хороший бизнес на изменении программ, написанных для одной платформы, для работы в другой системе. Такой процесс называется *переносом* (*porting*), или «портированием».

Как администратор FreeBSD, вы можете по-разному применять программы, написанные для другой платформы. Самый эффективный способ состоит в том, чтобы заново скомпилировать исходный код, сделав эти программы для FreeBSD «родными». Другие подходы, исключающие перекомпиляцию программы, подразумевают запуск «неродных» программ в эмуляторе или реализацию машинного (двоичного) интерфейса прикладных программ (Application Binary Interface, ABI) «родной» платформы.

Перекомпиляция

Многие программы в коллекции «портов» на самом деле представляют собой «родные» компилированные версии программ, разработанных для других платформ. Примерами могут служить KDE и Emacs. В сущности, про-

граммы, написанные для Linux, Solaris и других разновидностей UNIX, зачастую можно собрать из исходного кода (перекомпилировать). Чтобы эти программы без сучка без задоринки выполнялись во FreeBSD, требуется внести в их исходный код минимальные изменения, а порой можно обойтись и без таковых. Просто взяв исходный код и собрав его на машине FreeBSD, можно запускать эти программы во FreeBSD, словно «родные».

Перекомпиляция дает лучшие результаты, если платформы похожи. Например, FreeBSD и Linux предоставляют много идентичных системных функций: они собраны на стандартных функциях C согласно POSIX и используют сходные инструменты сборки и по большей части одинаковые системные вызовы.

Однако с годами различные платформы UNIX отошли друг от друга. В каждой версии UNIX реализованы новые функции, которым, в свою очередь, нужны новые библиотеки и функции. Если той или иной программе необходимы особые функции, ее не удастся собрать на других платформах.

Отчасти, стандарт POSIX был введен для смягчения этой проблемы. POSIX – это стандарт, описывающий минимально приемлемую систему UNIX и операционные системы, подобные UNIX. Программы, которые используют только POSIX-совместимые системные вызовы и библиотеки, могут быть сразу же перенесены на любые другие операционные системы, реализующие POSIX. Большинство производителей UNIX разрабатывают программное обеспечение в соответствии с POSIX.

Самое трудное – добиться того, чтобы разработчики следовали стандарту POSIX. Многие разработчики, создающие программы с открытым кодом, заботятся только о том, чтобы их программы запускались на их любимой платформе. Например, есть довольно много программ, специфичных для Linux, но не совместимых с POSIX. В свою очередь, код, полностью соответствующий POSIX, не может вобрать в себя преимущества специальных функций, предлагаемых той или иной операционной системой.

Например, во FreeBSD есть сверхэффективный системный вызов чтения данных, kqueue(2). Вместо него другие системы применяют select(2) и poll(2). Вопрос, который разработчики должны задать самим себе, таков: использовать kqueue, и тогда программа будет выполняться чрезвычайно быстро во FreeBSD и не будет работать в других системах, или применять select и poll, и тогда программа будет выполняться медленнее, но будет работать на других платформах. Разработчик может потратить больше времени на то, чтобы программа применяла различные функции на разных платформах. Это сделает счастливыми пользователей, но такой подход вызывает неприятие у разработчика. Какой бы вариант он ни избрал, кто-нибудь будет недоволен.

Проект FreeBSD выбирает средний путь. Если программу можно скомпилировать и запустить должным образом во FreeBSD, команда, разрабатывающая «порты», реализует такую возможность. Если программе необходимы незначительные заплатки, члены команды создают их и высылают разработчику этой программы. Большинство разработчиков программ охотно принимают

заплатки, что позволяет им поддерживать другие операционные системы. У них может не быть той или иной ОС для проведения тестов, или эта система может быть им неизвестна, однако если подходящая с виду заплатка приходит из уважаемого источника, она, вероятно, не будет отвергнута.

Эмуляция

Если для поддержки во FreeBSD программу пришлось бы значительно перестраивать либо исходный код просто недоступен, возможен другой вариант: *эмуляция*. Концепция эмулятора проста. Программа эмуляции преобразует системные вызовы одной операционной системы в системные вызовы, применяемые в локальной операционной системе. Программы, запускаемые в эмуляторе, работают, словно в «родной» системе. Однако преобразование системных вызовов означает дополнительные системные издержки, снижающие скорость выполнения программ в эмуляторе.

FreeBSD поддерживает широкое множество эмуляторов, большая часть которых представлена в коллекции «портов» (каталог */usr/ports/emulators*). В большинстве случаев эмуляторы полезны для обучения и развлечений. Если у вас есть неутолимое желание вновь запустить старую игру Commodore 64, можно установить */usr/ports/emulators/frodo*. (Также можно попробовать установить дисковод C64 и заставить его работать в UNIX. При этом вы узнаете о дисках больше, чем вам когда-либо хотелось, однако это совсем другая история.) Чтобы увидеть, как выглядят классические аппаратные средства UNIX, можно установить эмулятор PDP-11 в */usr/ports/emulators/sim*. (Полный список можно найти по адресу */usr/ports/emulators/README.html*.)

Однако поскольку эти эмуляторы на самом деле не нужны на сервере, здесь они не будут подробно рассматриваться. Но вы должны знать, что они доступны и где их можно найти.

Реализация ABI

В дополнение к перекомпиляции и эмуляции есть последний вариант, благодаря которому система FreeBSD снискала известность: реализация *машинного интерфейса прикладных программ* (Application Binary Interface, ABI). ABI – это часть ядра, которая предоставляет различные сервисы программам – от доступа к звуковым картам и чтения файлов до вывода изображения на экран и запуска других программ – все, в чем программы могут нуждаться. Если говорить о программах, то для них ABI – это операционная система. Полностью реализовав ABI той или иной операционной системы на своей системе, вы сможете выполнять «неродные» программы так, как они выполняются на «родной» платформе.

Хотя реализация ABI часто трактуется как «эмуляция», это не так. Предоставляя ABI, система FreeBSD не эмулирует системные вызовы, а реализует их естественным образом. Ко всему прочему, было бы неверным сказать, что «FreeBSD реализует Linux» или «FreeBSD реализует Solaris.» На самом деле, после того как эта технология была создана, никому не удалось одним словом

описать то, что проделала команда BSD. (Даже сегодня неясно, как это описать одним словом. Как же, черт возьми, *это* работает?) Чаще всего применяют термин «*режим*» (mode), например «режим Linux» или «режим osf1».

Трудности с эмуляцией ABI происходят из-за наложения имен. В большинстве операционных систем есть системные вызовы с общими именами, например read, write и т. д. Системный вызов read во FreeBSD работает совершенно иначе по сравнению с read в системе Windows. Если вы реализуете каждый системный вызов другой операционной системы в своей ОС, то ваша система будет реализацией другой ОС. Когда программа вызывает read, ей надо знать, какой это вызов – «родной» или «неродной». Системным вызовам можно дать различные имена, но тогда это будет нарушением стандарта POSIX. Либо надо предоставлять различные интерфейсы ABI и предписывать той или иной программе, какой ABI использовать. Именно такая схема реализована во FreeBSD.

Проверка типа исполняемого файла

Как правило, в операционных системах есть незамысловатая системная функция, которая запускает программы. Всякий раз, когда ядро отправляет программу модулю исполнения, модуль исполнения запускает программу.

Однако в какой-то момент в систему выполнения программ UNIX была добавлена специальная проверка. Если программа начинается с `#!/bin/sh`, то она передается командной оболочке, а не модулю исполнения. В BSD эта идея была логически продолжена: модуль исполнения взаимодействует с различными типами исполняемых файлов. Согласно своему типу, каждая программа перенаправляется соответствующему ABI. Таким образом, в системе BSD может быть несколько интерфейсов ABI, и, следовательно, она может поддерживать программы из различных операционных систем.

Отличительная черта такого механизма перенаправления – отсутствие издержек. Раз система так или иначе решает, как запускать программу, то почему бы не решить, какой ABI использовать? В конце концов, исполняемые файлы, созданные для различных операционных систем, имеют характерные черты, которые можно использовать для их идентификации; механизм идентификации и перенаправления делает этот процесс прозрачным для конечного пользователя.

Благодаря возможностям перенаправления программ разным ABI система FreeBSD способна запускать программы Linux, OSF/1, System V и SCO, как будто они были скомпилированы в ней самой. За счет этого значительно расширяется диапазон программ, способных работать под управлением FreeBSD.

Примечание

Такой диапазон интерфейсов ABI в системе FreeBSD поддерживается по двум причинам. Во-первых, тем, кто мог реализовать их, они были необходимы. Во-вторых, была доступна документация. Реализованные интерфейсы ABI очень похожи на системы FreeBSD; лишь несколько системных вызовов требуют серьезной доработки.

Какие интерфейсы ABI поддерживаются?

Описанная схема позволяет реализовать совершенно различные интерфейсы ABI. Например, кто-нибудь может взять информацию о Windows ABI, предоставляемую Microsoft, и написать модуль Windows ABI для FreeBSD, чтобы запускать в системе FreeBSD программы Windows естественным образом. Хотя это кажется привлекательным, для стабильной и надежной работы такого интерфейса потребовалось бы приложить нечеловеческие усилия. Это не было реализовано и, скорее всего, не будет. Наибольшая поддержка оказывается трем системам: SVR4, OSF/1 и Linux.

SVR4, или System V Release 4, была последним основным «выпуском» UNIX от AT&T. Она встречается в ранних версиях Solaris и SCO UNIX. В режиме SVR4 системы FreeBSD некоторые программы SCO выполняются быстрее, чем в самой SCO UNIX.

OSF/1, или Digital UNIX, была разработана для процессора Alpha. Сначала компания Digital создала процессор Alpha, а затем и Digital OSF/1. Поскольку OSF/1 использовала ядро Mach, а в состав FreeBSD не входят различные несовместимые с POSIX системные вызовы Mach, режим OSF/1 во FreeBSD реализован не полностью. Так или иначе, он не будет работать на аппаратных средствах, совместимых с Intel. (В отсутствие 64-разрядных команд на 32-разрядных машинах реализовать такой ABI довольно трудно!) Поскольку ваш процессор, скорее всего, не Alpha, здесь эта тема обсуждаться не будет.[1]

Наконец, режим *Linux* позволяет запускать программы Linux во FreeBSD. Этот ABI был протестирован наиболее тщательно, потому что исходный код Linux доступен, а ABI хорошо документирован. На самом деле режим Linux работает так хорошо, что многие программы в коллекции «портов» опираются на него. (Отдельные части этой книги были написаны с помощью StarOffice 5.2 в режиме Linux. Кроме того, я спокойно применял Linux Netscape и даже Linux WordPerfect.)

Библиотеки программ с других систем

Часть ABI, находящаяся в ядре FreeBSD, решает одну основную задачу. Однако есть и другие: в дополнение к средствам ядра каждой операционной системе необходимы свои возможности. Самый важный вопрос – совместно используемые библиотеки. Если ядро запускает программу, а программа не может найти библиотеки, она не будет работать правильно. Какой бы ABI вы ни использовали, вам нужны библиотеки этой платформы.

SVR4 и SCO

Например, для применения интерфейсов SVR4 и SCO необходим доступ к соответствующей системе. Для режима SVR4 будет достаточно компакт-диска

---

[1] Если у вас под рукой есть лишний компьютер с процессором Alpha (только не модели Multia, известной под названием Random Heat Death), можете переслать его мне под поручительство издательства No Starch Press; я с радостью рассмотрю режим OSF/1 в следующем издании книги.

Sun Solaris 2.6, однако для SCO ABI нужны библиотеки с действующей машины SCO UNIX. Следовательно, вам нужна лицензия SCO или Solaris. Конечно, это препятствие преодолимо, однако применять такой модуль уже сложнее и обойдется это дороже.

OSF/1

Минимальный набор библиотек OSF/1 есть в */usr/ports/emulators/osf1_base*. Эти библиотеки имеют ограничительную лицензию; их можно использовать далеко не всегда, однако при желании можно получить более полный набор библиотек с самой системы OSF/1. Если у вас есть лицензия OSF/1, с этими библиотеками можно делать все, что угодно.

Linux

Библиотеки для режима Linux – самые доступные. Поскольку барьер для доступа так невысок, совместимость с Linux здесь будет рассмотрена более подробно. Когда вы получите исчерпывающие знания о режиме Linux, вы сможете применить их для реализации любого ABI.

Установка и включение режима Linux

Проще всего установить и включить режим Linux с помощью «порта» */usr/ports/emulators/linux_base*, который загрузит и установит в */usr/compat/linux* большое подмножество типичных компонентов Linux. (Кроме того, в файл */etc/rc.conf* будет добавлена строка `LINUX_ENABLE="YES"`. Следовательно, модуль ядра, реализующий Linux ABI, будет запускаться при загрузке системы.)

В зависимости от того, какие программы установлены в системе, режим Linux может быть включен или выключен. Он работает настолько «прозрачно», что вы могли и не заметить, что тот или иной «порт» его установил! Чтобы выяснить, включен ли этот режим, проверьте по */var/db/pkg*, установлен ли пакет linux_base. Затем с помощью kldstat(8) посмотрите, загружен ли модуль ядра, реализующий Linux ABI:

```
# kldstat
Id Refs Address    Size     Name
 1    3 0xc0100000 236ff8   kernel
 2    1 0xc0337000 54f8     vesa.ko
 3    1 0xc119b000 12000    linux.ko
#
```

Как видите, последний модуль в предыдущем списке и есть модуль Linux ABI, linux.ko. Если модуль Linux не загружен, он не появится в списке.

Как показано в главе 4, если модуль действительно не загружен, его можно загрузить с помощью kldload(8):

```
# kldload linux
#
```

Чтобы модуль Linux автоматически загружался во время начальной загрузки системы, добавьте в */etc/rc.conf* такую строку:

```
LINUX_ENABLE="YES"
```

Теперь программы Linux должны работать без всякого дополнительного конфигурирования.

Идентификация программ

Современные исполняемые файлы UNIX представлены в формате ELF, который отводит место для комментария, или *маркировки* (*branding*). Программа будет передаваться тому или иному ABI в зависимости от марки. Если марки нет, этот исполняемый файл считается программой FreeBSD. FreeBSD распознает четыре марки: FreeBSD, Linux, Solaris и SVR4.

Хотя вы не можете непосредственно увидеть марку на исполняемом файле, марку можно узнать, а маркировкой манипулировать с помощью brandelf(1). Чтобы узнать марку того или иного исполняемого файла, просто запустите brandelf, указав имя файла в качестве аргумента:

```
# brandelf /bin/sh
File '/bin/sh' is of brand 'FreeBSD' (9).
#
```

Из вывода команды ясно, что это программа FreeBSD. Значит, она будет выполняться в интерфейсе FreeBSD ABI.

Если программа, написанная для другой системы, не запускается, проверьте ее марку. Отсутствие марки может объясняться следующим: FreeBSD пытается запустить эту программу в «родном» ABI. Изменить это можно ручной установкой марки с помощью `brandelf -t`. Например, чтобы установить марку Linux, выполните такую команду:

```
# brandelf -t Linux /usr/local/bin/program
#
```

При следующем запуске программы система попытается ее выполнить в Linux ABI. Если это программа Linux, можно ожидать положительного результата.

Что такое linux_base?

Модуль ядра, реализующий функции Linux, обеспечивает поддержку совместимости с Linux. Если вы решите применять ABI для другой системы UNIX, значительную часть этого интерфейса придется реализовывать самостоятельно. Таким образом, понимание работы режима Linux поможет справиться с неполадками в других режимах совместимости.

В первую очередь надо применять пакет linux_base, извлекающий часть установочного комплекта Red Hat Linux в */usr/compat/linux*. Содержимое этого каталога может выглядеть так:

```
#ls
bin     boot    etc     lib     mnt     opt     proc    sbin    usr     var
#
```

Похоже на содержимое корневого каталога, не правда ли? Да, если присмотреться внимательнее, можно отметить, что, вообще говоря, содержимое */usr/compat/linux* сравнимо с содержимым основной системы FreeBSD. Здесь можно найти те же программы, что и в базовой системе FreeBSD.

Примечание

Приверженец Linux сразу заметит, что содержимое linux_base лишь в минимальной степени сравнимо со списком программ, к которым он привык. В linux_base можно добавить другие программы Linux по своему усмотрению; об этом речь пойдет позже.

По возможности Linux ABI пытается «остаться» в */usr/compat/linux*. (Отчасти это похоже на незакрытую клетку.) Когда система выполняет программу Linux, которая вызывает другие программы, Linux ABI сначала ищет их в */usr/compat/linux*. Если в этом каталоге программа не найдена, то ABI ищет ее в основной системе FreeBSD.

Например, представим программу Linux, которая вызывает ping(8). Сначала ABI будет искать программу ping в */usr/compat/linux*. Обнаружив, что ее там нет, ABI обратится к основной системе FreeBSD и найдет */sbin/ping*.

А вот другая ситуация. Предположим, программа Linux хочет вызвать sh(1). Linux ABI сначала проверит */usr/compat/linux* и найдет *bin/sh*, а затем запустит ее вместо «родной» программы /bin/sh системы FreeBSD.

Добавление программ к linux_base

Как уже упоминалось, подсистема Linux в linux_base скорее минимальна. Некоторым программам Linux нужны дополнительные библиотеки. Разработчики FreeBSD стремятся к тому, чтобы «порты» были как можно меньше, однако в качестве компромисса дополнительные библиотеки Linux размещаются в дополнительных «портах». В состав коллекции «портов» входят несколько «портов», расширяющих linux_base. Большинство из них представляют собой совместно используемые библиотеки. Они позволяют увеличить список программ, которые поддерживаются в режиме Linux системы FreeBSD. В состав этих «портов» входят:

- devel/linux-libglade – библиотека графических интерфейсов, необходимая для работы различных программ;
- devel/linux_devel – коллекция инструментов для разработки программ Linux во FreeBSD;
- devel/linux_kdump – инструмент отладки программ Linux;
- graphics/linux-jpeg – содержит библиотеки для обработки файлов изображений JPEG;
- graphic/linux-png – устанавливает библиотеки для обработки файлов изображений PNG;

- graphic/linux-tiff – включает в себя библиотеки для обработки файлов изображений TIFF.

Установка этих «портов» обеспечит 99% функциональности, необходимой для запуска программ Linux.

Иногда может потребоваться библиотека или программа, которой нет в linux_base или коллекции «портов». В таком случае проще всего найти систему Linux соответствующей версии, скопировать необходимые файлы и установить их в */usr/compat/linux*. Если для библиотек вы создали новый каталог, внесите изменения в файл */usr/compat/linux/etc/ld.so.conf*.

Конфигурирование библиотек Linux

Метод конфигурирования библиотек, реализованный во FreeBSD, значительно отличается от метода, применяемого в Linux. В системе Linux из текстового конфигурационного файла создается файл кэша, подобный файлу */usr/compat/linux/etc/ld.so.conf*:

```
# cat ld.so.conf
/usr/i486-linux-libc5/lib
/usr/X11R6/lib
#
```

Первые символы косой черты в этих путях вводят в заблуждение. Почему? Вспомните, что Linux ABI ищет файлы в */usr/compat/linux first* и, если находит, то применяет их. На самом деле это каталоги */usr/compat/linux/usr/i486-linux-libc5/lib* и */usr/compat/linux/usr/X11R6/lib*. В этих каталогах находятся библиотеки Linux, устанавливаемые пакетом linux_base. Модуль ядра, реализующий Linux ABI, запускает программу Linux ldconfig(8) для чтения этого файла, а затем сканирует указанные каталоги и создает или обновляет */usr/compat/linux/etc/ld.so.cache*.

При добавлении библиотек к подсистеме Linux кэш необходимо обновить. Можно выгрузить и снова загрузить модуль ядра, реализующий функции Linux, однако это прервет работу службы. Другой подход состоит в том, чтобы запустить программу обновления кэша:

```
# /usr/compat/linux/sbin/ldconfig
#
```

Программа не выдаст на экран никаких результатов, однако можно проверить дату файла *ld.so.cache* и убедиться, что кэш был обновлен. Если дата – текущая, то так оно и есть.

Для упорядочивания добавляемых библиотек можно создать для них отдельный каталог. Например, я часто создаю */usr/compat/linux/usr/local/lib* для всякой всячины, которую я хочу добавить в систему. Однако для применения этих библиотек я должен добавить соответствующий путь к *ld.so.conf*. (Не забудьте отбросить первую часть пути – */usr/compat/linux*!)

Установка дополнительных пакетов Linux как RPM

Если программа Linux не может найти необходимую программу, ее надо добавить в */usr/compat/linux*. Режим Linux системы FreeBSD основан на Red Hat Linux, поэтому можно просто взять соответствующие компоненты Red Hat Linux и установить их в подсистеме Linux.

Red Hat Linux распространяется в формате менеджера пакетов Red Hat (Red Hat Package Manager, RPM). (Хороший выбор RPM-пакетов Red Hat Linux предоставляется на FTP-зеркалах по всему миру, а последний список зеркал можно найти на сайте *www.redhat.com*.) Файлы RPM подобны пакетам исполняемых файлов FreeBSD – они представляют собой сжатые архивы, в которых содержится все необходимое для запуска программы. RPM-пакеты устанавливаются и деинсталлируются как одно целое. Специалисты спорят о достоинствах RPM, pkg_add и многих других систем управления пакетами, однако пакет совместимости с Linux основан на Red Hat Linux, поэтому будет применять инструменты Red Hat.

Для использования RPMs необходимо установить соответствующие программы в */compat/linux*. Если бездумно запустить RPM согласно инструкциям на странице документации rpm(8), можно перезаписать часть системы FreeBSD. Это плохо – хотя во FreeBSD можно выполнять программы Linux, программы userland систем FreeBSD и Linux нельзя комбинировать произвольно. Такая попытка с большой вероятностью приведет к знакомству с процессом аварийного восстановления, описанным в главе 3.

Для надежной установки RPM выполните такую команду:

```
# rpm -i --ignoreos --dbpath /var/lib/rpm --root /compat/linux packagename
```

Примечание

Конечно, пакеты RPM совершено отличаются от обычных пакетов FreeBSD. Их нельзя удалить с помощью pkg_delete; для работы с ними необходимо применять RPM.

Использование нескольких процессоров – SMP

Компьютеры с несколькими процессорами существуют несколько десятков лет, но популярными они становятся сейчас, в эпоху Intel. Во FreeBSD поддержка нескольких процессоров обеспечивается начиная с версии 3, однако аппаратные средства для ее реализации только сегодня становятся по карману небольшим компаниям и энтузиастам.

Что такое SMP?

Симметричная многопроцессорная обработка (Symmetric MultiProcessing, SMP) подразумевает систему с несколькими (больше одного) идентичными процессорами. Предвосхищу ваш вопрос: да, существуют другие варианты организации многопроцессорной системы. Некоторые специалисты по вычислительной технике считают асимметричную многопроцессорную обра-

ботку более эффективной. Впрочем, вы не можете купить такие аппаратные средства, следовательно, спорить не о чем.

SMP обладает многими преимуществами по сравнению с отдельными процессорами, и дело здесь не в обычном «больше мощности!». Если рассуждать на микроуровне, процессор за раз может выполнять только одну операцию. Процессы в компьютере конкурируют за время процессора. Если процессор выполняет запрос к базе данных, он не примет пакет, который пытается доставить карта Ethernet. Каждую долю секунды процессор *переключает контекст* и работает с другим процессом, назначенным ядром. Такое переключение происходит довольно часто и достаточно быстро, поэтому кажется, что одновременно выполняются несколько операций – подобно тому, как телевизионная картинка изменяется за счет очень быстрой смены отдельных кадров. При наличии нескольких процессоров компьютер может выполнять несколько операций одновременно. Это замечательно, однако данная возможность чрезвычайно увеличивает сложность системы.

Поскольку один процессор может выполнять единовременно только одну операцию, во многих программах был предусмотрен обход этого ограничения. На самом деле многие программы представляют собой более одного процесса. К примеру, веб-сервер Apache фактически запускает довольно много процессов, которые обслуживают веб-страницы, а значит, хорошо подходит для работы на многопроцессорных системах.

Долгое время многопроцессорная обработка была отличительной чертой коммерческих систем UNIX. Только что Sun Microsystems анонсировала систему SPARC со 102 процессорами. Даже Windows 2000 поддерживает несколько процессоров – в какой-то мере бестолковым образом. Однажды я привез домой 4-процессорную систему Intel 486, и, хотя руки до нее так и не дошли, что-то мешает мне ее выбросить. Сегодня множество производителей предлагают системные платы X86 SMP, в том числе такие именитые компании, как Dell и Compaq.

Допущения о работе ядра

Для понимания многопроцессорной обработки и проблем, связанных с ней, необходимо углубиться в работу ядра. Все операционные системы, в которых реализуется SMP, сталкиваются с одними и теми же проблемами. Следовательно, теория, представленная здесь, подходит для различных платформ. Однако FreeBSD в какой-то мере отличается от других операционных систем, поскольку опирается на 30-летнее наследие UNIX, а разработка FreeBSD не прекращается ни на месяц.

Следующее описание представляет собой грубое упрощение. Конструкция ядра – мудреная тема. Почти невозможно описать ее точно на уровне, который поймут непрограммисты. Вот объяснение работы ядра в самом общем виде.

Со стороны кажется, что компьютер выполняет несколько операций одновременно. Например, у меня запущен WindowMaker, Netscape оживленно загружает данные через кабельный модем, плюс продолжается сборка «порта».

Обрабатываются сетевые прерывания, экран отображает новый текст, веб-сервер Apache выдает страницы и т. д. На самом деле одновременность этих операций – кажущаяся. Единовременно средний процессор может выполнять что-то одно.[1]

Во FreeBSD использование процессора организовано по принципу *квантов времени* (*slices*). *Квант* – это период времени, который процессор тратит на одну задачу. Один процесс может использовать процессор в течение всего кванта либо до того момента, когда задача будет выполнена. После этого может быть запущен другой процесс. Для выделения квантов времени и их распределения между программами ядро применяет систему приоритетов. Если в системе появляется процесс, приоритет которого выше приоритета текущего процесса, ядро прерывает обработку первого процесса, или *выгружает* его. Обычно такой режим называют вытесняющей многозадачностью.

Идем дальше. Если ядро запущено, это еще не процесс. Ядро запускает процессы. Процесс имеет определенные структуры данных, задаваемые ядром, и ядро управляет ими так, как считает нужным. Ядро можно представить как особый процесс, сильно отличающийся от других процессов. Он не может быть прерван другими программами – нельзя набрать `killall kernel` и перезагрузить систему. Кроме того, обычно ядро не проявляется в выводе top и других подобных инструментов.

В старых ядрах разных UNIX и, в частности, FreeBSD трудности с SMP преодолеваются так: ядро объявляется непрерываемым – его работа не может быть прервана. За счет этого управление ядром становится простым и ясным: если какой-либо модуль ядра занимает участок памяти, он может рассчитывать на данный участок при выполнении следующей команды. Никакой другой модуль ядра не может захватить эту память.

Ситуация изменилась (к лучшему) после версии 2.2.

FreeBSD 3.0 SMP

Первая реализация SMP в FreeBSD была довольно проста. Процессы распределялись между процессорами с грубым соблюдением баланса нагрузки. Для ядра существовала специальная «блокировка» (lock). Для выполнения кода ядра процессор должен был захватить «блокировку», а перед этим проверить, доступна ли она. Если «блокировка» была доступна, он захватывал ее и запускал ядро. Если же «блокировка» была недоступна, процессор знал, что ядро выполняется другим процессором, и переключался на обработку чего-то другого. Такая «блокировка» была названа *Big Giant Lock* (BGL). В такой системе ядро знало, что его данные не затронет ни один процесс. По существу,

[1] Некоторые процессоры (Alpha) могут одновременно выполнять несколько операций. Процессоры, в которых одновременно выполняются две или четыре команды, медленно набирают популярность. Это одна из причин, по которым процессор Alpha представлял собой такую замечательную технологию и почему так жаль, что Alpha уже нет.

данная схема гарантировала, что ядро будет запущено только на одном процессоре, как это всегда и было.

Такая стратегия была довольно эффективна в случае двух процессоров. Можно было запустить базу данных среднего уровня и веб-сервер и быть уверенным, что подсистема процессоров не станет причиной снижения производительности. Если один процессор был занят обслуживанием веб-страниц, то другой мог обрабатывать запросы к базам данных. Однако на машине с 8 процессорами вас поджидали трудности. Значительную часть времени система ждала, пока освободится BGL! Ядро по-прежнему знало, что оно единовременно выполняет только одну операцию. Если команда ядра изменяла какое-то внутреннее значение, это по-прежнему было так, как если бы это значение возвращалось. В такой системе есть много подводных камней, но в целом она проста – не эффективна и не масштабируема. По существу, в руководствах по SMP этот метод упоминается редко, поскольку он довольно неуклюж. Тем не менее он превосходит методы организации SMP, предлагаемые другими системами. Например, по умолчанию в двухпроцессорной системе Windows 2000 один процессор выделяется для пользовательского интерфейса, а другой – для всего остального. И хотя интерфейс получается быстрый, а мышь не замирает при загрузке системы, я надеюсь, что большинство пользователей не станут приобретать аппаратные средства SMP для решения проблем с графическим интерфейсом.

В ближайшие годы с развитием аппаратных средств многопроцессорные системы найдут широкое распространение. Для того чтобы FreeBSD продолжала быть качественной операционной системой, трудности с SMP необходимо преодолеть.

FreeBSD 5 SMP

Одним из преимуществ слияния BSDi/Walnut Creek стала передача кода BSD/OS 5.0 сообществу разработчиков FreeBSD. BSD/OS содержит значительный объем патентованной информации, поэтому исходный код нельзя передать во всеобщее пользование. Тем не менее разработчики FreeBSD смогли изучить часть кода. Наиболее интересная особенность данного кода заключалась в том, что ядро могли одновременно обслуживать несколько процессоров. В значительной степени такая возможность была реализована в версии 5, что стало одним из главных различий между 4-й и 5-й версиями FreeBSD.

Чтобы предотвратить искажение данных, в новых SMP-системах FreeBSD комбинируется Big Giant Lock и маленькая блокировка, называемая *семафором* (*mutex*).[1] Когда тот или иной модуль ядра хочет поработать с участком данных, он устанавливает на них семафор. Когда другой модуль ядра пытается обратиться к этому участку, он говорит себе : «Я не могу получить

[1] Поскольку «семафором» в UNIX называется еще один механизм взаимодействия процессов, не связанный с SMP, в русскоязычной литературе для обозначения таких «SMP-семафоров» часто употребляют англицизм «мьютекс». – *Примеч. науч. ред.*

эти данные», а дальше либо ждет, пока ресурс станет доступным, либо обращается к другому ресурсу. Цель этой схемы – исключить Big Giant Lock и добиться того, чтобы все операции ядра устанавливали семафоры-блокировки только на те данные, которые им необходимы. Когда подсистемы ядра будут переписаны в расчете на преимущества семафоров, необходимость в BGL отпадет. По словам Грега Лиэя (Greg Lehey), ключевого разработчика FreeBSD и участника проекта SMP, этот метод позволит применять свыше 32 процессоров.

Примечание

Блокировку BGL можно убрать полностью и заменить на семафоры одним движением недрогнувшей руки (как делают производители коммерческих ОС), потратив на это пару месяцев. Почему бы не поступить именно так? Да потому, что тогда версия FreeBSD-current была бы совершенно непригодной в течение нескольких месяцев, а 5.0-release был бы плохо отлажен. Кроме того, разработчикам-добровольцам, создающим другие части системы, было бы нечего делать. (Сказать волонтерам, что им нечего делать, – значит, потерять их.)

Представленная информация о механизмах SMP позволяет достаточно хорошо управлять ими. Теперь рассмотрим подробно, как обслуживать SMP-систему.

Применение SMP

Применяя SMP, помните, что несколько процессоров необязательно ускоряют выполнение операций. Один процессор может выполнять определенное количество операций в секунду; наличие второго процессора означает, что компьютер может выполнять в два раза больше операций в секунду, однако эти операции не обязательно выполняются быстрее.

Проведем аналогию процессоров с полосами движения на дороге.[1] Если на дороге одна полоса, за один раз финишную ленту может пересечь одна машина. Если на дороге четыре полосы, эту ленту могут одновременно пересечь четыре машины. Хотя дорога с четырьмя полосами вовсе не обязательно позволит этим машинам быстрее добраться до пункта назначения, многие из них придут к финишу в одно время. Если вы не чувствуете разницу, представьте, что произойдет, если кто-нибудь перекроет вам путь на дороге с одной полосой. Пропускная способность процессорной подсистемы очень важна.

Большинство пользовательских процессов не должны заботиться о том, когда применять SMP; процесс лишь запрашивает процессорное время, а ядро его выделяет. Программе все равно, откуда берется время CPU.

Сложности с SMP возникают, когда требуется, чтобы один процесс обрабатывался на нескольких процессорах. Краткое пояснение: это не удастся, если программа не многопоточная. Многопоточные программы написаны спе-

[1] Этот пример предполагает, что каждый водитель не превышает скорость, соблюдает очередь и не «подрезает» другие машины – другими словами, не поступает так, как поступают водители в любом американском городе.

циально для выполнения на нескольких процессорах. (Посмотрите в документации, является ли программа многопоточной.) Программы, запускающие множество процессов для обслуживания запросов (например, Apache), не являются многопоточными, но могли бы ими быть. В целом, Apache замечательно использует преимущества нескольких процессоров.

SMP и обновления

Самая распространенная «проблема», которая возникает в связи с SMP, проявляется при выполнении самого мучительного теста – обновления из исходного кода. Кажется, что система никогда не использует более одного процессора одновременно. Программа «top» показывает, что 50% ресурсов системы не занято, и все тут.

Поверьте своим глазам. Если система наполовину бездействует, она использует только один из процессоров. Утилита make, которая выполняет сборку программ, выполняет команду, ждет отклика, а затем выполняет другую команду. Каждую из подзадач можно назначить разным процессорам, но make не приступит к следующему этапу, пока предыдущий не выполнится. Одновременно выполняется только одна операция.

Обойти эту проблему можно с помощью ключа `-j` программы make, который предписывает ей задействовать несколько процессоров одновременно. У ключа `-j` есть свой аргумент – количество запускаемых процессов make:

```
# make -j4 buildworld
```

Данная строка предписывает запускать четыре процесса. Надо надеяться, что теперь сборка завершится быстрее. Однако это не означает четырехкратное ускорение работы make; остаются другие трудности, которые надо преодолеть (см. главу 10).

Примечание

Не все программы можно собирать с ключом `-j`. Не всегда получается даже buildworld. (Ведется дискуссия об отключении поддержки `make -j` в buildworld, поскольку она вызывает много проблем.) Ключ `-j` стоит попробовать, однако если дела пойдут плохо, надо вернуться к последовательному выполнению команд make.

Несколько процессоров – это не самое важное в достижении высокой производительности. Приложения должны быть написаны с учетом преимуществ многопроцессорной обработки. В противном случае дополнительные процессоры не принесут пользы.

12

Поиск хостов с помощью DNS

Система доменных имен (Domain Name Service, DNS) – одна из тех незаметных, действующих за кулисами программ, которым не уделяется и половины того внимания, которого они заслуживают. Многие пользователи никогда не слышали о DNS, однако именно эта система делает Интернет таким, каким мы его знаем. DNS, или *служба имен*, обеспечивает отображение между именами хостов и IP-адресами. Без DNS веб-броузеру и программам электронной почты не были бы понятны удобные имена, подобные *www.cnn.com*, и пришлось бы набирать IP-адреса, что значительно снизило бы популярность Интернета.

Система DNS необходима любым сервисам Интернета. Здесь мы обсудим работу DNS, способы его проверки, конфигурирование системы FreeBSD для использования DNS и сборку собственного DNS-сервера.

Как работает DNS

DNS просто отображает IP-адреса в имена хостов, а имена хостов – в IP-адреса. Например, пользователю не надо знать о том, что *www.AbsoluteBSD.com* на самом деле представляет собой последовательность чисел 209.69.178.25. Пользователь хочет просто набрать URL в своем веб-броузере и получить искомую страницу. Преобразование имя–адрес выполняет DNS. Системный администратор должен уметь устанавливать, контролировать и проверять информацию DNS, а также понимать, как система выполняет эти операции.

Информация DNS может находиться в любом месте: в локальной системе, на локальном DNS-сервере, на удаленном сервере имен. В системах UNIX для предоставления такой информации используется распознаватель (resolver) – программа,[1] которая знает обо всех источниках данной информации и взаимодействует с ними. Когда программе требуется узнать IP-адрес хоста или имя хоста по IP-адресу, она обращается к распознавателю. Распознаватель

---

[1] Если не упрощать схему слишком сильно, надо понимать, что resolver – это не отдельная программа ОС, а функция системной библиотеки, которая прикомпоновывается к программе при сборке. – *Примеч. науч. ред.*

консультируется с соответствующими источниками информации и возвращает результат программе, которой он необходим. Далее в этой главе будет рассмотрена настройка распознавателя.

Как правило, DNS-запрос, получаемый от программы, распознаватель перенаправляет серверу имен – компьютеру, на котором работает программа, собирающая информацию DNS с других компьютеров в Интернете. Когда к серверу имен поступает DNS-запрос, сервер имен обращается к своему локальному кэшу и смотрит, запрашивалась ли эта информация в последнее время. (Серверы имен получают много идентичных DNS-запросов; например, сервер имен одной компании-провайдера услуг Интернета, с которой я сотрудничал, в течение часа получал несколько сотен запросов IP-адреса для *www.cnn.com*. Добавьте сюда запросы к Yahoo!, eBay и MSN, и станет ясно, что кэш довольно эффективен.)

Если на означенном сервере имен нет необходимой информации, он обращается к *корневому серверу* (*root server*), который хранит список серверов имен, отвечающих за каждый домен в Сети. В процессе, называемом *рекурсивным запросом* (*recursive query*), корневой сервер предписывает серверу имен обратиться к подходящим серверам имен, которые, в свою очередь, могут перенаправить запрос другим серверам. В конце концов запрос попадает к серверу имен, отвечающему за этот домен, и первоначальный сервер имен получает ответ на свой вопрос.

Примечание

Регистрируя домен, необходимо указать два сервера имен. У хостов должна быть возможность обратиться к этим серверам и получить данные о домене. Если один сервер имен не работает, второй должен выдерживать всю нагрузку, а если не действуют оба сервера имен, домен «исчезает» из Интернета. Тогда при попытке обратиться к вашему домену (*www.yourdomain.com*) пользователь получит сообщение об ошибке «домен не найден». Письма будут возвращаться отправителям. Все будут считать, что вы не существуете. Такое иногда происходит даже с большими компаниями, такими как Microsoft. Менеджер или заказчик уведомит вас о сбое, причем сделает это не самым приятным образом. Уделяйте внимание своей службе имен!

Основные инструменты DNS

В состав FreeBSD входят несколько инструментов для проверки информации DNS. Поскольку большинство серверов DNS работают по протоколу передачи дейтаграмм пользователя (User Datagram Protocol, UDP), вы не можете применить telnet и непосредственно обратиться к серверу, как в случае с электронной почтой и веб-сервисом (о них речь пойдет позже). Доступ к информации DNS можно получить только с помощью host(1) и dig(1).

Команда host

Для быстрого получения IP-адреса хоста применяйте команду host(1). Например, адрес веб-страницы моего издателя можно получить так:

```
# host www.nostarch.com
www.nostarch.com is a nickname for nostarch.com
nostarch.com has address 66.80.60.21
nostarch.com mail is handled (pri=20) bysmtp.lax.megapath.net
nostarch.com mail is handled (pri=10) by mail.nostarch.com
#
```

Эта информация вызывает интерес, поскольку из нее следует, что, согласно DNS, у хоста есть несколько имен и несколько веб-страниц с одним IP-адресом. Основная веб-страница No Starch Press на самом деле является псевдонимом для другого имени хоста, *nostarch.com*; один IP-адрес может иметь любое количество имен. Такая схема подобна семейной телефонной системе, в которой несколько человек «разделяют» один телефон. В этой аналогии люди соответствуют именам хостов, а телефонный номер – IP-адресу. У хоста есть одно каноническое имя – точно так же телефон зарегистрирован на одного человека.

Примечание

Многим серверным программам требуется намного больше, чем просто IP-адрес. Например, если в веб-броузере вы введете *http://66.80.60.21*, то, как ни странно, окажетесь на странице провайдера, размещающего у себя сайт No Starch Press, а не на странице самого издательства No Starch Press. В главе 15 описывается, как популярные веб-серверы размещают множество веб-сайтов на одном IP-адресе, однако на будущее надо запомнить, что такая возможность существует.

Получение подробной информации с помощью dig

Команда host довольно полезна, но она, безусловно, не предоставляет подробную информацию. Кроме того, вы не знаете, откуда взялись данные – взяты ли они из кэша или же сервер имен добыл их на сервере, ответственном за этот домен. Программа dig(1) – это стандартная программа для нахождения подробной информации DNS. (Другой инструмент, nslookup(1), был популярен многие годы, но потом вышел из моды.) У dig есть множество ключей, которые позволяют выявлять различные неполадки, связанные со службой имен. Здесь будут рассмотрены базовые возможности этой программы.

Базовая форма ее вызова включает имя «dig» и имя хоста. Например, для того чтобы найти сведения о моем издателе, можно ввести такую команду:

```
# dig www.nostarch.com
❶ ; <<>> DiG 8.3 <<>> www.nostarch.com
❷ ;; res options: init recurs defnam dnsrch
❸ ;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: ❹ NOERROR, id: 4
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2
❺ ;; QUERY SECTION:
;;      www.nostarch.com, type = A, class = IN
❻ ;; ANSWER SECTION:
www.nostarch.com.       2h13m2s IN CNAME  nostarch.com.
nostarch.com.           2h13m2s IN A    66.80.60.21
```

```
❼ ;; AUTHORITY SECTION:
nostarch.com.           7h48m45s IN NS  NS1.MEGAPATH.NET.
nostarch.com.           7h48m45s IN NS  NS2.MEGAPATH.NET.
❽;; ADDITIONAL SECTION:
NS1.MEGAPATH.NET.       7h48m35s IN A   216.200.176.4
NS2.MEGAPATH.NET.       7h48m35s IN A   216.34.237.2
;; Total query time: 11 msec
;; FROM: blackhelicopters.org to SERVER: default -- 127.0.0.1
;; WHEN: Sun Apr  7 12:24:17 2002
;; MSG SIZE  sent: 34  rcvd: 144
```

Ого! Столько информации! (Скорее всего, вы применяете dig для отладки чего-либо. Лучше получить больше информации, чем столкнуться с ее нехваткой.)

Итак, что мы выяснили? Прежде всего, строки, начинающиеся точкой с запятой, являются комментариями. Либо в них перечислены ключи, используемые программой dig, либо они разбивают ответы на разделы.

Первая строка (❶) сообщает версию программы dig и ключи, заданные в командной строке. Во второй строке (❷) представлены ключи, которые применяет dig. (Так как в командной строке ключи не были указаны, dig применяет ключи по умолчанию; некоторые полезные ключи будут обсуждаться далее.)

Третья строка (❸) сообщает, где программа dig получила ответ (если получила). Следующая строка (❹) содержит важное слово, `NOERROR`, которое говорит о том, что dig нашла ответ, кажущийся подходящим. Если слова `NOERROR` нет, что-то не так. Типичные ошибки: `NXDOMAIN` (домен не существует) или `SERVFAIL` (домен на сервере неверно сконфигурирован).

Следующая пара строк содержит коды, которые на самом деле нужны лишь при отладке DNS и для особых операций. Поэтому перейдем к следующим четырем разделам: query (запрос), answer (ответ), authority (авторитетный источник) и additional section (дополнительный раздел).

Раздел query

В разделе `query` (❺) видно, как dig интерпретирует запрос. В данном примере раздел query выглядит так:

```
;;      www.nostarch.com, type = A, class = IN
```

Начнем с самого простого – с класса (`class`). Хотя DNS может обслуживать различные системы имен, нас интересует система Интернета, или `IN`. Домены Интернета должны всегда иметь класс `IN`.

Тип (`type`) – это тип записи, которую мы ищем. По умолчанию dig ищет запись `A` (address, адрес): есть имя, нужен IP-адрес. Запись `PTR` (pointer, указатель) означает, что по IP-адресу требуется найти соответствующее имя.

Раздел answer

Следующий раздел – answer (❻). Первая строка содержит имя хоста, адрес которого мы ищем (*www.nostarch.com*). Следующий элемент (`2h13m2s IN CNAME`) сообщает время жизни этой информации – период времени, в течение кото-

рого локальный сервер имен может ее кэшировать. Выражение `2h13m2s` означает, что данные потеряют силу через 2 часа 13 минут и 2 секунды. Напомним, что `IN` указывает на данные Интернета. Наконец, `CNAME` означает, что далее следует *каноническое имя* этого сервера. Данная запись говорит о том, что наш домен на самом деле является псевдонимом (alias) для чего-то еще. Вывод dig подтверждает, что *www.nostarch.com* на самом деле является псевдонимом для *nostarch.com*.

Вторая строка раздела answer почти идентична первой, за исключением того, что вместо `CNAME` в ней расположен символ «A». Такая запись говорит о том, что далее следует IP-адрес.

Раздел authority

Для получения IP-адреса *www.nostarch.com* необходимо пройти по цепочке авторитетных серверов имен. В разделе authority (❼) перечислены серверы имен, ответственные за домен, их время жизни, а также данные и серверы, которые они представляют. В нашем случае серверами имен являются `ns1.megapath.net` и `ns2.megapath.net`. Данные для *www.nostarch.com* будут находиться в кэше в течение 7 часов 48 минут и 45 секунд. Это снова записи Интернета (IN). NS означает «серверы имен» (nameservers).

Раздел additional

Наконец, в разделе additional (❽) dig перечисляет IP-адреса всех хостов, представленных вместе с интересующим нас хостом. В данном примере перечислены серверы имен для *www.nostarch.com*: *ns1.megapath.net* и *ns2.megapath.net*. Вызывает интерес тот факт, что время жизни здесь составляет 7 часов 48 минут и 35 секунд. В этом случае время жизни означает не период времени, в течение которого локальный сервер имен должен хранить информацию о хостах, а период времени, в течение которого сервер имен, ответственный за *nostarch.com*, хранит о нем информацию в своем кэше. Когда это время истечет, ваш сервер имен отбросит эту информацию и затребует список серверов имен с одного из корневых серверов имен Интернета. Это необязательно плохо или хорошо, однако при возникновении неполадок об этом механизме полезно знать.

Запустите dig несколько раз с разными доменами и разберитесь с выводом этой команды.

Поиск имен хостов с помощью dig

Предположим, что IP-адрес известен, и надо выяснить связанное с ним имя хоста (желание, аналогичное желанию выяснить владельца телефонного номера). Это типичная задача в Интернете – пользователь с тем или иным IP-адресом может обращаться к вашему веб-серверу каждые пять секунд, а вы хотите узнать, кто он такой и что пытается сделать.

Для нахождения имени можно применить *обратный поиск* (*reverse lookup*) с помощью ключа `-x` команды dig. Поскольку большая часть вывода практи-

чески идентична результатам прямого поиска, рассмотрим только тот раздел, в котором есть различия: answer.

```
# dig -x 66.80.60.21
;; ANSWER SECTION:
21.60.80.66.in-addr.arpa.  2h24m IN PTR  www.megapathdsl.net....
#
```

Мы знаем, что на этом IP-адресе находятся по крайней мере *www.nostarch.com* и *nostarch.com*, обратный поиск покажет истинное имя хоста. На самом деле эти два хоста располагаются на машине с именем `www.megapathdsl.net`.

Обозначение PTR в середине строки свидетельствует о том, что это запись-*указатель*. Это специальный вид записи, используемый для выделения записей «IP-адрес/имя хоста». Когда вы будете устанавливать DNS, каждому хосту с IP-адресом потребуются записи `PTR` и `A`.

Дополнительные ключи dig

Программа dig понимает широкое множество ключей командной строки, позволяющих управлять поиском информации. Для получения полного списка ключей обратитесь к странице руководства dig(1). Там рассмотрены ключи, позволяющие непосредственно задавать имя сервера имен и отменить рекурсию. Эти ключи применяются чаще всего.

Имя сервера

Первый ключ – имя сервера. По умолчанию dig запрашивает первый сервер имен, указанный в */etc/resolv.conf*. Однако при устранении неполадок может возникнуть потребность обратиться к другим серверам имен. Для этого при запуске dig следует указать в командной строке ключ `@`. Например, чтобы спросить у *dns1.yahoo.com*, что он знает об *AbsoluteBSD.com*, введите такую команду:

```
# dig @dns1.yahoo.com AbsoluteBSD.com
```

Вывод этой команды двольно похож на вывод, представленный в предыдущем примере, за исключением того, что вы увидите ссылки на сервер имен Yahoo!. При выявлении неполадок необходимо тщательно сравнить эту информацию с результатами, выдаваемыми локальным сервером имен. Если данные от двух разных серверов имен противоречивы, это может послужить ключом к разгадке. (Информация DNS должна изменяться лишь при добавлении, переименовании и перенумерации хостов.) Для точной локализации проблемы можно применить другие ключи dig.

Управление выполнением запроса

Программа dig имеет ключи двух типов: одни ключи управляют ее запуском, а другие – выполнением запросов. Ключи, управляющие запуском, предваряются знаком «минус» (-); ключи, управляющие тем, как dig выполняет запросы, предваряются знаком «плюс» (+). В dig есть много дейст-

вительно хитроумных приемов, но они выходят за рамки этой главы. Впрочем, возможность управлять выполнением запросов довольно-таки полезна.

По умолчанию для получения ответа сервер имен выполняет рекурсивные запросы. Хотя спросить сервер имен о домене полезно, при отладке порой нужна не информация о домене, а поочередная проверка каждого сервера имен в цепочке. Для этого можно применить ключ `norecurse`:

```
# dig AbsoluteBSD.com +norecurse
```

Попробуйте указать домен, к которому сервер имен никогда не обращался. Моим любимым сайтом для тестирования DNS-сервера является *www.moo.com* – просто потому, что звучит здорово, таинственно, притягательно и, в общем-то, безобидно, и туда никто не обращается. С достаточной степенью уверенности можно предположить, что ни один пользователь, обращающийся к вашему серверу имен, в последнее время не заглядывал на *moo.com*.

Если эта информация не кэшировалась сервером имен либо срок ее достоверности истек и она удалена из кэша, то первый раздел вывода будет похож на результаты предыдущего примера. Однако раздел authority будет выглядеть иначе:

```
;; AUTHORITY SECTION:
com.                    18h9m59s IN NS  A.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  G.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  H.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  C.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  I.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  B.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  D.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  L.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  F.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  J.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  K.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  E.GTLD-SERVERS.NET.
com.                    18h9m59s IN NS  M.GTLD-SERVERS.NET.
```

Серверы имен в `GTLD-SERVERS.NET` – это корневые серверы. Они содержат мастер-списки, в которых сказано, какой сервер имен управляет тем или иным доменом. Выдавая такой результат, ваш сервер имен говорит: «Ну, я не знаю. Я собирался спросить кого-нибудь еще, но мне запретили выполнять рекурсию, поэтому я здесь и остановился.»

Для обращения к корневому серверу имен одновременно задайте ключ `norecurse` и укажите имя сервера:

```
# dig @a.gtld-servers.net +norecurse www.moo.com
```

Пройдитесь по цепочке серверов имен для одного-двух сайтов, и вы на самом деле поймете, как работает DNS.

Настройка клиента DNS: распознаватель

Прежде чем обращаться к DNS-серверу в своей системе, вы должны сообщить компьютеру, какой сервер и как следует использовать. Установка клиентской части службы имен необходима даже DNS-серверу, потому что компьютер не узнает о том, что на нем запущен сервер имен, если ему об этом не сказать! Работающий клиент службы имен необходим почти для всех операций в сети. С помощью ключевых слов в */etc/resolv.conf* можно проинформировать распознаватель системы о том, куда он должен обращаться в поисках информации.

Ключевые слова domain или search

Работая на машинах в своей локальной сети, вы не хотите набирать полные имена хостов. (Если у вас 30 веб-серверов, то, набирая `ssh www19.mycompany.com` и ожидая ответ, можно состариться.) Для того чтобы указать распознавателю, какие домены проверять по умолчанию, можно в */etc/resolv.conf* применить ключевые слова `domain` или `search`.

Задание локального домена

Ключевое слово `domain` говорит распознавателю, в каком домене по умолчанию искать хосты. Например, чтобы задать *AbsoluteBSD.com* в качестве локального домена, введите такую строку:

```
domain     AbsoluteBSD.com
```

После того как задан локальный домен, любая команда, которой обычно требуется доменное имя, будет считать, что доменом является *AbsoluteBSD.com*. Если я выполняю команду `ping www`, распознаватель добавит имя «AbsoluteBSD.com» к «www» и предпишет выполнять `ping www.AbsoluteBSD.com`.

Задание списка доменов с помощью search

В качестве альтернативы я могу применить ключевое слово `search` и указать список доменов. Возможно, моей компании принадлежат несколько доменов в различных частях сети. В таком случае я могу ввести следующую строку:

```
search AbsoluteBSD.com blackhelicopters.org stenchmaster.org
```

Тогда распознаватель будет проверять эти домены в порядке их написания, пока не найдет совпадение.

Например, если я выполняю `ping petulance`, распознаватель попытается найти *petulance.AbsoluteBSD.com*. Если найти не удастся, он проверит *petulance.blackhelicopters.org* – следующий домен по порядку. Наконец, он проверит *petulance.stenchmaster.com*. Если такого хоста нет ни в одном из этих доменов, команда не будет выполнена.

Примечание

Если вы не укажете ни одно из ключевых слов `domain` и `search`, распознаватель будет использовать доменное имя локальной машины.

Список серверов имен

Итак, распознаватель знает, в каких доменах искать по умолчанию, и надо сказать ему, к каким серверам имен обращаться. Перечислите каждый сервер имен в отдельной строке – в порядке предпочтения. Согласно этому порядку распознаватель будет обращаться к серверам имен. Список выглядит примерно так:

```
nameserver 127.0.0.1
nameserver 209.168.70.3
nameserver 192.168.87.3
```

Заметим, что первая запись в списке – IP-адрес 127.0.0.1, принадлежащий «обратной петле» (loopback). Данная запись нужна на машине, выполняющей функции сервера имен, поскольку она предписывает распознавателю обращаться к серверу имен локального хоста. Иногда, довольно редко, возникает желание не применять локальный сервер. Можно поступить и так, но в большинстве случаев это приводит к напрасной трате сетевой полосы пропускания.

Если в */etc/resolv.conf* представлены записи `nameserver`, а также ключевые слова `domain` или `search`, файл готов.

Источники информации DNS

Для того чтобы управление сервисами Интернета было эффективным, необходимо управлять своей доменной службой имен. Большинство провайдеров предоставляют такой сервис, но кто захочет согласовывать со специалистами провайдера важные изменения в своей инфраструктуре?

Теперь вы знаете, как выглядят данные DNS и как работает цепочка авторитетных серверов DNS, и можете создать свой сервер имен. В состав FreeBSD входят все программы, необходимые для сервера DNS. Надо только настроить их и запустить. Подготовим два источника, из которых будем черпать информацию об именах хостов и их IP-адресах: *файл hosts* и *демон named*. Конфигурируют их по отдельности.

Файл hosts

В файле */etc/hosts* сопоставляются IP-адреса и имена отдельно взятых хостов. Однако, хотя файл *hosts* очень прост, его содержимое имеет силу только на отдельной машине. Файл одной системы нельзя применять на другой без дополнительных ухищрений.

Динамические серверы имен в значительной степени вытеснили */etc/hosts*, однако файл *hosts* все еще полезен в маленьких сетях и за устройством преобразования сетевых адресов (Network Address Translation, NAT). Например, файл *hosts* хорошо подходит, если у вас есть один или два сервера, а кто-нибудь другой отвечает за управление вашей службой имен, доступной извне. Если же у вас несколько серверов, каждый из которых должен поддерживаться отдельно, необходим полноценный сервер имен.

Каждая строка в */etc/hosts* представляет один хост. Первая запись в каждой строке – это IP-адрес, а вторая – *полностью определенное доменное имя* (*fully qualified domain name*) хоста, например *mail.mycompany.com*. Далее можно перечислить все псевдонимы этого хоста.

Например, в небольшой компании может быть один сервер, который обрабатывает почту, предоставляет сервисы FTP, веб, DNS и выполняет множество других функций. Рабочая станция в данной сети может иметь такую запись в */etc/hosts*:

```
192.168.1.2     mail.mycompany.com     mail ftp www dns dns
```

Воспользовавшись этой записью, рабочая станция может найти этот хост (*mail.mycompany.com*) либо по полному доменному имени, либо по кратким псевдонимам (*ftp.mycompany.com*, *www.mycompany.com* и т. д.).

Если вы обнаружите, что вам необходимо более двух или трех строк в *hosts* или поддерживать файлы *hosts* становится трудно, это сигнал к тому, что для обслуживания данных о хостах надо создать сервер имен. Сервер имен намного более масштабируем по сравнению с файлами *hosts* на каждой машине. Обслуживать сервер имен намного проще – стоит только его установить.

Демон named

Самое популярное программное обеспечение для создания сервера DNS – это BIND (Berkeley Internet Name Daemon).[1] Сама серверная программа называется named(8). На самом деле BIND представляет собой набор инструментов, в состав которого входят named и вспомогательные программы, такие как dig.

BIND поддерживается Консорциумом программного обеспечения Интернета (Internet Software Consortium, *www.isc.org*) и выпускается под лицензией, подобной лицензии BSD. Хотя у BIND есть конкуренты, например djbdns (*/usr/ports/net/djbdns*), BIND считается эталонной реализацией службы имен, поэтому здесь будет рассматриваться именно этот пакет. Концепции, применяемые в BIND, в целом подходят для других серверов имен.

Последние несколько лет BIND был объектом атак злоумышленников, поэтому его полностью переписали, при этом основное внимание было уделено безопасности. В самой последней версии реализованы очень мощные функции защиты, а при разработке применялась технология «безопасного программирования».[2]

Первичный и вторичный серверы DNS

Независимо от того, какой демон сервера имен вы изберете, необходимо понимать, что такое первичный (*master*) и *вторичный* (*slave*) серверы DNS. Для

---

[1] Разработчики BIND расшифровывают эту аббревиатуру иначе – вместо Daemon пишут Domain. См. *http://www.isc.org/products/BIND/*. – *Примеч. науч. ред.*

[2] Defensive programming – «безопасное программирование» – технология предупреждения потенциальных ошибок путем проверки множества допустимых условий в каждом модуле. – *Примеч. перев.*

обслуживания каждого домена требуется по крайней мере два сервера имен. Только один из них может быть первичным, все остальные – вторичные.

Первичный сервер имен – последняя авторитетная инстанция для домена. Информацию о домене надо изменять на первичном сервере имен, так как именно оттуда эту информацию берет вторичный сервер.

Один сервер имен может быть первичным для одних доменов и вторичным для других. Например, *AbsoluteBSD.com* имеет два сервера имен – *blackhelicopters.org* и *ralph.glblnet.com*. При этом первый хранит оригинальные файлы описания этого домена, и все изменения выполняются на этой системе. Таким образом, *blackhelicopters.org* является первичным сервером имен. Время от времени *ralph.glblnet.com* обновляет свои записи для этого домена, беря информацию с *blackhelicopters.org*. Значит, *ralph.glblnet.com* является вторичным сервером. Если систему *blackhelicopters.org* похитят инопланетяне, то *ralph.glblnet.com* продолжит предоставлять информацию DNS для *AbsoluteBSD.com*.

С другой стороны, *ralph.glblnet.com* хранит мастер-записи для многих других доменов, а другие серверы имен обновляют свои записи для этих доменов, обращаясь к *ralph.glblnet.com*. Следовательно, *ralph.glblnet.com* является одновременно первичным и вторичным сервером имен, но для разных доменов.

Прямые и обратные DNS

Возможно, вы слышали или сталкивались с концепциями *прямого* (*forward*) и *обратного* (*reverse*) DNS. Прямой DNS подразумевает определение IP-адреса по имени хоста. Примеры прямого DNS можно найти в A-записях примеров с dig:

```
nostarch.com.           2h13m2s IN A    66.80.60.21
```

Буква A означает, что это адресная запись, или прямой DNS. Она так и называется: «A-запись» или «адресная запись».

Обратный DNS предполагает получение имени хоста по IP-адресу. Предположим, системный протокол свидетельствует, что кто-то пытается подключиться к серверу SSH с IP-адреса 66.80.60.21, и нужно определить имя хоста. Для этого можно воспользоваться ключом `-x` программы dig. Большей частью вывод похож на вывод прямого поиска, но ответ существенно иной:

```
# dig -x 66.80.60.21
…
;; ANSWER SECTION:
21.60.80.66.in-addr.arpa.  2h24m IN PTR www.megapathdsl.net.
…
#
```

Исторически сложилось так, что IP-адреса отображаются в обратном порядке как часть домена in-addr.arpa (в случае обратного поиска). Далее следуют обычные данные о времени жизни и запись IN, обозначающая данные Интернета.

Особый интерес вызывает PTR-запись, или *запись-указатель*. Она сообщает о том, что IP-адрес «указывает» на имя. По существу, это каноническое, наиболее точное имя данного IP-адреса. Проведем аналогию с телефонной системой: несколько человек совместно используют телефонный номер, зарегистрированный на кого-то одного.

По идее, прямые и обратные DNS должны совпадать, но несколько хостов могут разделять один IP-адрес, и поэтому A-записи вовсе не обязательно нужна соответствующая PTR-запись. Например, мы видели раньше, что хост *nostarch.com* имеет IP-адрес 66.80.60.21, однако с данным IP-адресом ассоциировано имя *www.megapathdsl.net*. Соответствие должно выдерживаться для A-записи. Если имя хоста, заданное при обратном поиске, не имеет соответствующей прямой записи, то это означает, что DNS сконфигурирован неверно, а инструменты, полагающиеся на информацию DNS, скажем, TCP Wrappers, будут отклонять запросы на подключение с этих систем. К счастью, проверить прямой и обратный DNS можно с помощью автоматизированных инструментов.

in-addr.arpa

С PTR-записями связано одно существенное затруднение: часто они выводятся в обратном порядке.

Итак, DNS сличает хосты слева направо. При анализе хоста *www.AbsoluteBSD.com* сервер имен сначала ищет сервер имен для *.com*. Далее в домене *.com* проводится поиск *AbsoluteBSD.com*, а затем в *AbsoluteBSD.com* обнаруживается *www.AbsoluteBSD.com*. Самые крупные элементы находятся слева, однако в IP-адресе они располагаются справа. Для анализа IP-адреса его необходимо развернуть наоборот. Например, адрес 66.80.60.21 преобразуется в 21.60.80.66.

Спутать прямой и обратный IP-адреса довольно легко, поэтому DNS использует специальный маркер для обозначения обратного адреса. В конце обратного IP-адреса стоит строка «in-addr.arpa». (Причины такого расположения уходят корнями в прошлое; они довольно скучны и здесь рассматриваться не будут.) Главное, адрес 66.80.60.21 станет 21.60.80.66.in-addr.arpa.

Но почему нельзя оставить IP-адрес таким, какой он есть, и не использовать in-addr.arpa для обозначения того, что это обратный DNS? Хорошо, что вы спросили. Предыдущий адрес прост: если запустить dig, он будет проверять очень узкое пространство. Однако в случае большой сети необходимо выполнить DNS-запрос, анализирующий значительно больший диапазон IP-адресов. Например, 118.168.192.in-addr.arpa будет преобразован в адреса сети 192.168.118. Может даже потребоваться 168.192.in-addr.arpa и 192.in-addr.arpa. Каждый такой запрос подразумевает проверку значительного пространства – аналогично запуску `dig .com`. (Вероятно, вам никогда не понадобится запускать `dig .com`, однако это потребуется инженерам, обслуживающим магистрали Интернета. Именно они и пишут такие программы. Один из недостатков профессиональных инструментов заключается в том, что в первую очередь они адресованы профессионалам.)

Примечание

Если нужна быстрая и приближенная информация DNS, программа host(1) сама выполнит обратное преобразование. То же самое сделает программа dig, если запустить ее с ключом -x. Однако не смущайтесь, увидев запись in-addr.arpa.

Конфигурирование Named

Перед запуском демон named необходимо сконфигурировать. Каталог */etc/namedb* содержит основные файлы настройки named.

named.root

Файл *named.root* должен присутствовать обязательно, но редактировать его не нужно. В нем перечислены *корневые серверы имен*. Если сервер имен получает запрос информации о сайте, которой нет в его кэше, он обращается к корневым серверам имен. (Этот файл изменяется нечасто – последний раз он обновлялся в августе 1997 года.) Редактировать данный файл надо лишь в том случае, если система не подключена к Интернету, а использует частный корневой сервер.

named.conf

Другой важный файл – *named.conf*. Это основной файл named. Если *named.conf* содержит ошибки, сервер имен неработоспособен.

Синтаксис файла *named.conf* напоминает код на языке C. Однако не переживайте, если языка C вы не знаете. Правила очень просты, а примеры объясняют все, что вам необходимо знать. Строки, начинающиеся с двух символов косой черты (//), представляют собой комментарии. А любой текст между старыми добрыми знаками комментариев C (/* и */) – это многострочный комментарий.

В *named.conf* есть два типа записей: *options* и *zones*. Каждая запись в конфигурационном файле должна быть либо параметром, либо зоной, либо комментарием. *Зона* – это причудливое название домена (строго говоря, они не идентичны, но для наших целей можно сделать такое допущение). Параметры управляют работой BIND.

Параметры

Если не принимать во внимание комментарии, файл *named.conf*, поставляемый по умолчанию, начинается со списка опций. Основная их часть неясна и по умолчанию закомментирована. Для применения параметров их необходимо разместить в соответствующем разделе, который ограничен словом «options» и фигурными скобками. Сами параметры располагаются между скобками и отделяются друг от друга точкой с запятой. Вот очень простой раздел options из файла *named.conf*:

```
options {
        directory "/var/named";        listen-on { 127.0.0.1; 209.69.178.18; } ;
} ;
```

В этом примере параметр `directory` имеет значение `"/var/named"`, а в параметре `listen-on` перечислены два IP-адреса.

Сначала рассмотрим параметр `directory`. Он задает каталог, в котором *named.conf* ищет и хранит файлы DNS. Если начать с этого параметра, дальнейшая настройка сервера будет более простой.

Каталог, задаваемый по умолчанию (*/etc/namedb*), вполне приемлем, если вы организуете службу имен для двух доменов. Однако если DNS предоставляется для десятков и сотен клиентов, этот каталог быстро наполнится и раздела root ему станет недостаточно.

Стандартная альтернатива каталогу */etc/namedb* – каталог */var/named*. Здесь хранятся файлы службы имен на больших серверах. Обычно я выбираю */var/named*, даже если надо обслуживать несколько доменов, так как эти файлы непрерывно растут.

Параметр `listen-on` определяет, на каких IP-адресах named принимает запросы. Если за одной сетевой картой закреплены десятки IP-адресов, named можно привязать только к одному адресу. (Такое решение особенно ценно, если в системе применяются клетки.)

BIND поддерживает намного больше параметров, но эти два, вероятно, применяются чаще других. Полный список параметров и описание их применения можно найти в руководстве операторов BIND (BIND Operators Guide) на сайте *http://www.isc.org*.

Зоны

По умолчанию *named.conf* описывает три зоны, или домена, которые сервер имен обслуживает по умолчанию: *корневую зону* (*root zone*), *локальную зону* для Ipv4 (IPv4 localhost) и *локальную зону для Ipv6* (IPv6 localhost). Каждой из этих зон соответствует запись в *named.conf*, следующая за списком параметров. Зоны, представленные по умолчанию, настраивать не надо – решение об их изменении наверняка будет ошибочным. Однако мы рассмотрим, для чего они предназначены.

Корневая зона

Сервер имен обращается к корневой зоне, когда у него нет информации о запрашиваемом домене или хосте. Такие запросы перенаправляются корневому серверу имен. Вот запись в *named.conf* для корневой зоны:

```
zone ❶ "." {
        ❷ type hint;
        ❸ file "named.root";
};
```

Первый элемент (❶) говорит, для какого домена предназначена эта запись. Точка в кавычках означает, что запись соответствует всему Интернету.

Поле `type` (❷) определяет тип домена. Корневая зона – особая: только она имеет тип `hint`.

Наконец, ключевое слово `file` (❸) сообщает демону named, в каком файле хранится информация для этого домена. Демон обращается к каталогу, указанному в параметре `directory`, находит файл с этим именем и назначает его содержимое данной зоне. Эти файлы будут рассмотрены позже.

Локальные зоны

Локальные (localhost) зоны (IPv4 и IPv6) применяются для локального хоста; они предоставляют сервисы DNS для IP-адреса обратной петли, 127.0.0.1. Без них каждый системный вызов, запрашивающий имя локального хоста, переходил бы в режим ожидания, существенно замедляя работу системы. Каждая из этих зон подобна корневой зоне, но имя файла другое.

Вот конфигурация для локальной зоны IPv4. В файле *named.conf* она расположена сразу после корневой зоны:

```
zone ❶ "0.0.127.IN-ADDR.ARPA" {
        ❷ type master;
        ❸ file "localhost.rev";
} ;
```

Довольно похоже на блок `options` и корневую зону из предыдущего раздела, не правда ли?

Имя зоны (❶) представлено в кавычках после слова `"zone"`. Поскольку эта зона применяется для обратного DNS, мы видим `IN-ADDR.ARPA`. (Если данный IP-адрес записать в обратном порядке, получится фактическая группа IP-адресов 127.0.0.)

Поле `type` (❷) указывает, является ли данный сервер первичным или вторичным для этого домена. Каждый сервер имен является первичным для локальных зон.

Наконец, ключевое слово `file` (❸) сообщает серверу имен, в каком файле можно найти информацию об этом домене. Информация об этой зоне находится в файле *localhost.rev*, расположенном в каталоге, указанном в параметре `directory`.

Настройка вторичного сервера имен

Возможно, самая легкая задача при установке DNS – настроить вторичный сервер имен. Запись выглядит аналогично записям для корневой и локальной зоны. Надо знать имя домена, для которого данный сервер имен будет вторичным, и IP-адрес первичного сервера имен.

Для настройки вторичного сервера имен скопируйте запись локальной зоны и немного измените ее. К примеру, конфигурация для вторичного сервера, обслуживающего домен *AbsoluteBSD.com*, очень похожа на корневую и локальные зоны.

```
❶ zone "AbsoluteBSD.com" {          ❷ type slave;                ❸ file "
AbsoluteBSD.com.db";                ❹ masters { 209.69.178.18;} ;
}
```

Есть имя домена (❶), тип зоны (❷) и имя файла (❸). В указанном файле хранится информация для данного домена. По традиции имя файла образуется добавлением к имени домена расширения *.db*. (Вопреки ожиданиям, данный файл не является базой данных.) Он создается, когда вторичный сервер загружает данные о домене с первичного сервера.

Далее идет IP-адрес первичного сервера (❹). Вторичный сервер запрашивает информацию о домене с первичного сервера через постоянные интервалы времени. (Интервалы будут рассмотрены позже.) Первичный сервер имен должен быть представлен IP-адресом; в конце концов, у DNS-сервера должна быть возможность загружать свои записи еще до того, как он узнает IP-адрес чего-либо!

Настройка первичного сервера имен

Конфигурация *named.conf*, необходимая для первичного сервера, даже проще конфигурации для вторичной зоны:

```
❶ zone "AbsoluteBSD.com" {          ❷ type master;                ❸ file "
AbsoluteBSD.com.db";
}
```

Итак, еще раз: есть имя домена (❶), тип зоны (❷) и имя файла (❸). В отличие от файла описания домена для вторичного сервера имен, данный файл надо создать. Создание этого файла рассматривается в разделе «Файлы зон».

Настройка нескольких зон

Если вы управляете крупными серверами имен Интернета, вы несете ответственность за тысячи доменов. Сделаете ошибку – и гнев множества людей обрушится на вас. Поэтому перед настройкой сотен зон подумайте о том, как их разместить.

Упростить настройку нескольких зон поможет отделение файлов зон, для которых сервер является первичным, от файлов зон, для которых сервер является вторичным (резервным). Обычно я создаю два каталога, *master* и *slave*. Файлы в каталоге *master* священны, поэтому их надо беречь. Файлы для вторичной зоны не являются мусором, но потерять их не страшно.

Если необходимо обслуживать тысячи доменов, разбиение файлов первичных зон можно продолжить. В моем каталоге *master* 36 каталогов – по одному для каждой буквы и цифры. Конечно, можно создать иерархию каталогов по своему усмотрению. Но помните, что будете либо работать с этой иерархией, либо с досадой изменять ее.

Запись о зоне, доведенная до логического завершения, может выглядеть так:

```
zone "AbsoluteBSD.com" {
        type master;
        file "master/clients/a/absolutebsd.com";
} ;
```

Чаще всего в таком количестве каталогов необходимости нет, но если потребуется, иерархию можно создать.

Файлы зон

Итак, у нас есть конфигурационный файл, который сообщает демону named, за какие домены он отвечает и где расположены файлы с информацией об этих доменах. Однако эти файлы еще надо создать!

Синтаксис файлов зон не отличается понятностью, поскольку, подобно sendmail, BIND создавался программистами, которые были более заинтересованы в эффективности, чем в простоте использования. В отличие от sendmail, конфигурирование файлов зон не настолько сложно для пользователя, однако отдельные части файлов зон кажутся противоречивыми.

Для того чтобы изучить работу с файлами зон, ознакомьтесь с представленными примерами. И когда будете удивленно почесывать затылок, глядя на их содержимое, помните, что пробираетесь через первобытное болото Интернета. (Если бы DNS был изобретен сейчас, файлы зон, вероятно, выглядели бы совсем иначе.)

Вот простой пример конфигурирования файла зоны. В состав FreeBSD входит сценарий командной оболочки `make-localhost` для создания файла *localhost*. Надо просто перейти в каталог */etc/namedb* и выполнить такую команду:

```
# sh make-localhost
```

И все! Файл *localhost.rev* создан. Это первый пример, поэтому разберем данный файл подробно.

```
;       From: @(#)localhost.rev 5.1 (Berkeley) 6/30/90
; $FreeBSD: src/etc/namedb/PROTO.localhost.rev,v 1.6 2000/01/10 15:31:40 peter Exp
$
;
; This file is automatically edited by the `make-localhost' script in
; the /etc/namedb directory.
;
❶ $TTL    3600
❷ @       ❸ IN      ❹ SOA     ❺ satariel.blackhelicopters.org.
❻ root.satariel.blackhelicopters.org.  (
                                ❼ 20010601        ; Serial
                                ❽ 3600    ; Refresh
                                ❾ 900     ; Retry
                                ❿ 3600000 ; Expire
                                ⓫ 3600 )  ; Minimum
        IN      NS      satariel.blackhelicopters.org.
1       IN      PTR     localhost.blackhelicopters.org.
```

Прежде всего, вспомним, что все строки, начинающиеся с точки с запятой, представляют собой комментарии. (Тщательно комментируйте файлы зон; впоследствии эта информация поможет вспомнить, почему настройка выполнялась так, а не иначе.)

$TTL: время жизни

Пропустим комментарии и перейдем к первой строке, представляющей интерес – оператору `$TTL` (❶). В нем предопределяется время жизни зоны, в секундах (в данном случае 3600 секунд). Это значение определяет, как долго другие серверы будут кэшировать информацию об этой зоне. Данным оператором можно определить любое время жизни зоны по своему усмотрению. На самом деле предпочтителен короткий промежуток; хорошее среднее значение – 10 800 секунд, или 3 часа. Выбор TTL сродни черной магии; значение по умолчанию подойдет в большинстве случаев.

Start of Authority

Следующая запись – Start of Authority (SOA, начало авторитетности). В ней представлено краткое описание зоны – каково ее поведение и как серверам следует себя с ней вести. Каждая зона имеет только одну запись SOA. В нее входит не информация о домене, а лишь сведения о продолжительности ее хранения.

Знак @

Символ «at» (`@`), с которого начинается SOA-запись (❷), – это специальное сокращение для «зоны, указанной в *named.conf*». В данном случае *named.conf* говорит, что в этом файле хранятся данные для зоны 0.0.127.in-addr.arpa. Когда named считывает *named.conf* и загружает его содержимое в память, он выполняет такую замену. Фактическое доменное имя было бы более понятным для новичков, однако такая запись встречается в большинстве серверов имен и, следовательно, ее необходимо понимать. Вместо символа @ можно применять полное доменное имя, но почти никто так не поступает.

Тип данных и обозначение

`IN` представляет тип данных (❸) – данные Интернета. `SOA` означает запись Start of Authority (❹). Оба элемента присутствуют в каждой создаваемой записи DNS.

Имя машины

Следующая часть – имя машины, на которой находится мастер-файл (❺). (Этот файл создан на *satariel.blackhelicopters.org*.)

Ответственная сторона

Далее следует электронный адрес человека, ответственного за эту зону (❻). Поскольку сценарий make-localhost по умолчанию задает адрес *root@hostname*, в нем отсуствует знак @, потому что этот знак соответствует названию зоны, указанному в *named.conf*.[1] (Если оставить @ в адресе электронной почты, то адрес превратился бы в *root0.0.127.in-addr.arpa.satariel.blackhelicopters.org*. Это хуже, чем *root.satariel.blackhelicopters.org*, не

[1] DNS был создан еще до того, как знак @ стал широко применяться в электронных адресах. Такое перекрытие – порок электронной почты, а не BIND.

правда ли?) Такую особенность следует учитывать при создании файлов зон для своих доменов. Знак @ в электронном адресе заменяется точкой.

В большинстве случаев на сервере имен нет почтового сервера. Согласно установившейся в Интернете практике, в поле электронного адреса поставьте hostmaster и свое доменное имя. Предполагается, что каждый домен имеет электронный адрес «hostmaster», предназначенный для ответа на вопросы, связанные с DNS.[1]

Круглые скобки

Формально запись SOA должна располагаться в одной строке. Однако тогда ее было бы трудно читать. Поэтому в стандартных файлах зон эта запись разбита на несколько строк. Открывающая круглая скобка указывает на обрыв строки. Каждая строка до закрывающей круглой скобки считается частью записи SOA.

Порядковый номер

Порядковый номер (`Serial`) обозначает версию файла зоны (❼). Порядковый номер может быть любым, однако удобнее всего применять дату. Обычно она записывается в формате YYYYMMDD с двумя дополнительными цифрами в конце. Последовательный номер в листинге, 20010601, означает 1 июня 2001 года. Две дополнительные цифры сообщают, сколько раз файл изменялся в течение дня. Мне, бывало, приходилось обновлять информацию о домене по десять раз в день. При этом каждый раз надо было обновлять значение порядкового номера. Механизм изменения таков. Скажем, я создаю файл зоны 9 мая 2002 года и задаю порядковый номер 2002050901. Если я меняю файл зоны 8 июня, порядковый номер изменится на 2002060801. Если я изменю файл зоны второй раз в этот же день, порядковый номер будет 2002060802. Такая система допускает до 100 изменений в день – примерно одно изменение каждые 15 минут. Если окажется, что этого недостаточно, то надо переосмыслить рабочий процесс.

Порядковый номер важен, поскольку время от времени вторичный сервер связывается с первичным и смотрит, не обновилась ли зона. Для этого он сравнивает порядковый номер в своем кэшированном экземпляре зоны с порядковым номером файла зоны на первичном сервере. Если порядковый номер на первичном сервере больше, чем на вторичном, вторичный сервер определяет, что файл зоны был обновлен, и загружает новейшую информацию о домене.

Примечание

Если вторичные серверы имен не обновили свои файлы зон с первичного сервера имен, причиной могут быть неполадки с порядковым номером. Даже если вы ручаетесь, что ин-

[1] Только не забудьте настроить sendmail так, чтобы письмо на этот адрес до вас действительно дошло, поскольку на него будет выслана информация при регистрации домена; забудьте об этом, если настраиваете домен, который не будет виден из Интернета. – *Примеч. науч. ред.*

кременти́ровали порядковый номер, инкрементируйте еще раз и посмотрите, что получится. Возможно, все встанет на свои места.

Refresh

Следующее число – значение `Refresh` (обновлять), в секундах (❽). Оно определяет, как часто вторичные серверы связываются с первичным и проверяют, не обновился ли мастер-файл. Согласно значению `Refresh` в файле *localhost.rev*, вторичный сервер имен будет обращаться за обновленным мастер-файлом каждые 3600 секунд, или 60 минут.

Если вторичный сервер не может сравнить свои данные с данными на первичном сервере, в ответ на запросы он выдает текущую информацию из файла зоны, – в конце концов, для этого и предназначен резервный сервер! Данный механизм будет подробно рассмотрен в разделе «Refresh, retry и expire на практике».

Retry

Следующее число – значение `Retry` (повторная попытка), также в секундах (❾). Если вторичный сервер не может связаться с первичным, он предпримет повторную попытку спустя этот период времени. В данном примере значение `Retry` составляет 900 секунд (15 минут). Если вторичный сервер имен не сможет обновить свои данные через час, он будет пытаться сделать это каждые 15 минут, пока первичный сервер имен не отзовется. Данный механизм будет подробно рассмотрен в разделе «Refresh, retry и expire на практике».

Expire

Далее следует значение `Expire` (истечение срока действия), в секундах (❿). Если сервер имен не может обновить свои записи в течение этого периода времени, он прекращает выдачу кэшированной информации о зоне. По мнению администратора, в данных обстоятельствах наличие недействительной информации хуже ее отсутствия. В нашем примере период времени составляет 3 600 000 секунд (1000 часов, или чуть более 41 дня).

Minimum TTL

Последнее число – минимальное время жизни (⓫). В старых реализациях BIND это значение определяло время жизни буквально всей информации. Сейчас оно означает только TTL для отрицательных ответов. (Серверы имен могут кэшировать негативные ответы.) Например, если вы ищете *givememymoneyback.AbsoluteBSD.com*, ваш сервер имен узнает, что такого хоста не существует. В соответствии с параметрами в *localhost.rev*, отрицательные ответы для этой зоны будут кэшироваться в течение 3600 секунд (1 час).

Примечание

Убедитесь, что после значения времени жизни стоит закрывающая круглая скобка! В противном случае named предположит, что остаток файла также представляет собой часть SOA-записи, и будет сбит с толку.

Повторное использование записей SOA

Итак, вы получили полное представление о записи SOA, и для вас есть приятная новость. Создав одну запись SOA по своему усмотрению, вы можете повторно использовать ее для других доменов, внося в нее минимальные изменения. На основе шаблона SOA я настроил тысячи доменов.

Информация о домене

Вы сформировали полную запись SOA и можете внести фактическую информацию о домене. Она расположена сразу же после записи SOA.

В следующем примере две строки содержат информацию о фактических хостах зоны:

```
        IN      NS      satariel.blackhelicopters.org.
1       IN      PTR     localhost.blackhelicopters.org.
```

Каждая строка состоит из четырех частей: имя хоста или число, тип данных, тип сервера и фактические данные. Первое поле содержит либо имя хоста (например, *www*), либо число (например, 12). Имя зоны автоматически присоединяется либо в начало, либо в конец этой записи – в зависимости от того, для чего предназначен этот файл – для обратного или прямого DNS. Поскольку наш образец предназначен для обратного DNS, цифра 1 добавляется к 127.0.0. В результате получается IP-адрес 127.0.0.1. (Если в первом поле ничего нет, named так или иначе добавит имя зоны. В результате получится разумное значение по умолчанию.) Тип данных всегда `IN` (Интернет).

Третье поле, тип сервера, представляет особый интерес. NS-запись (показана в первой строке примера) представляет сервер имен. В нашем примере единственным сервером имен для данного домена является *satariel.blackhelicopters.org*. Если вы размещаете *localhost.rev* на нескольких серверах имен, в них надо добавить дополнительные строки NS.

Примечание

Первое поле строки NS, представленной в примере, ничем не заполнено, поэтому BIND считает, что это зона 0.0.127.in-addr.arpa или сеть, начинающаяся с 127.0.0, то есть домен, указанный для этого файла в *named.conf*.

`PTR`-запись (показана во второй строке предыдущего примера) представляет отображение «IP-адрес/имя хоста». Эта зона – для сети 127.0.0. Во второй строке примера в первом поле стоит .1. Такая запись представляет .1 в сети, или 127.0.0.1, стандартный IP-адрес обратной петли. Запись указывает на локальный хост – в нашем случае это *localhost.blackhelicopters.org*.

Refresh, Retry и Expire на практике

Уже говорилось о том, что означают параметры `Refresh`, `Retry` и `Expire`, теперь посмотрим, как они работают. Вот пример, показывающий их в действии.

Предположим, у нас есть домен с периодом `Refresh`, равным 4 часам, периодом `Retry`, равным 1 часу, и периодом `Expire`, равным 48 часам. Это означает,

что вторичный сервер имен связывается с первичным ежечасно и проверяет наличие обновлений. Время от времени вы редактируете записи на первичном сервере имен, и в течение часа они передаются на вторичный сервер.

Пока что все хорошо. Теперь предположим, что первичный сервер имен взрывается, а осколки жесткого диска разлетаются по трем округам. Что произойдет с вторичным сервером? В следующий раз, когда вторичный сервер попытается заполучить обновленные записи, он не сможет связаться с первичным сервером. В этот момент он изменяет периодичность своих обращений за обновлениями. Вместо периода `Refresh` (4 часа) вторичный сервер будет применять время `Retry` (1 час). Как только он сможет успешно проверить статус записей, он вернется к использованию значения `Refresh` вместо значения `Retry`.

Если вторичный сервер не может подтвердить свои данные в течение периода `Expire`, информация о домене будет признана устаревшей и ненужной. В этом случае вторичный сервер отбросит данные и будет возвращать ошибку в ответ на любой запрос об этом домене. Домен исчезнет из Сети.

Если ваш первичный сервер имен действительно исчезнет ужасным образом, у вас есть время, равное сумме значений `Expire` и `Retry`, чтобы заменить данный сервер или переконфигурировать вторичный сервер, сделав его первичным.

Пример реально существующей зоны

Файл локальной зоны отчасти надуман – он представляет только одну машину с одним IP-адресом. Однако он удобен, его можно найти на любом сервере имен, а данные, содержащиеся в нем, либо широко используются, либо совершенно необходимы.

Теперь рассмотрим файл зоны, который более полно представляет домены, находящиеся на обслуживании. Обсудим важные фрагменты из *named.conf* и файла зоны для *AbsoluteBSD.com*.

named.conf

Вот фрагмент из *named.conf*:

```
zone "AbsoluteBSD.com" {
        type master;
        file "master/AbsoluteBSD.com";
} ;
```

В этом примере мы сообщаем серверу named, что он отвечает за домен *AbsoluteBSD.com*, для которого он является первичным сервером имен. Также указано имя файла, в котором можно найти информацию о домене. Если параметр `directory` имеет значение `/var/named`, то этот файл можно найти в */var/named/master/AbsoluteBSD.com*. Рассмотрим его без лишних слов.

/var/named/master/absolutebsd.com

```
❶ $TTL 345600

❷ @       IN      SOA     blackhelicopters.org. root.blackhelicopters.org.  (
                               2001101501      ; Serial
                               86400   ; Refresh -- 24 hours
                               7200    ; Retry -- 2 hours
                               2592000 ; Expire -- 30 days
                               345600) ; Minimum -- 4 days

          ❸ IN      NS      blackhelicopters.org.
            IN      NS      ralph.glblnet.com.

          ❹ IN      MX    10    blackhelicopters.org.

            IN      MX    20    ralph.glblnet.com.

❺           IN      CNAME   www.AbsoluteBSD.com.
www         IN      A       209.69.178.30
```

Данный файл похож на файл *localhost*, однако это файл действительной зоны для реально существующего домена Интернета. Посмотрим, что в нем есть. Во-первых, есть время жизни (❶). Его значение равно четырем дням – когда сервер имен получает IP-информацию для этого домена, он хранит ее четыре дня. SOA-запись (❷) содержит контактную информацию и значения времени `Refresh`, `Retry` и `Expire`, а также порядковый номер.

В файле зоны перечислены два сервера имен (❸): `blackhelicopters.org` и `ralph.glblnet.com`. Согласно значениям времени в файле зоны, *ralph.glblnet.com* будет сравнивать свои записи с записями *blackhelicopters.org* каждые 24 часа. Если попытка сравнить записи завершится неудачей, то сравнение будет осуществляться каждые 2 часа. Если *ralph.glblnet.com* не может сравнить свои записи с записями первичного сервера в течение 30 дней, он перестает выдавать ответы на запросы о домене *AbsoluteBSD.com*. Наконец, удаленные серверы имен будут кэшировать ответы «нет такого хоста» в течение четырех дней.

Почтовый ретранслятор

Далее представлен новый тип записи, `MX` – *почтовый ретранслятор* (*mail exchanger*) домена (❹). Хотя домен имеет только один первичный почтовый хост, у него могут быть несколько резервных почтовых серверов. Тем не менее почта должна обязательно дойти до основного почтового хоста. В этих записях указываются предпочтительный почтовый и резервные серверы. (Эта тема будет подробно обсуждаться в главе 14.)

Приоритет

Дополнительное поле в записи `MX` – *приоритет* (*preference*) (числа 10 и 20). Серверы с меньшим приоритетом считаются более предпочтительными. В на-

шем случае сервер *blackhelicopters.org* с приоритетом 10 является предпочтительным почтовым сервером для *AbsoluteBSD.com*. Если *blackhelicopters.org* недоступен, *ralph.glblnet.com* выступит в качестве резервного сервера.

Когда-нибудь вы, возможно, захотите добавить еще один почтовый сервер к двум уже имеющимся или изменить предпочтительный сервер, поэтому между приоритетами необходимо предусмотреть определенное пространство. Если поступить иначе и нумеровать их по порядку – 1, 2, 3 и т. д., то такая схема окажется менее гибкой. Например, в тот день, когда на *AbsoluteBSD.com* почту получают тысячи клиентов, MX-записи будут выглядеть так:

```
IN      MX      10      mail.AbsoluteBSD.com
IN      MX      20      mail2.AbsoluteBSD.com
IN      MX      30      mail.someothercompany.com
```

Записи о хосте

Наконец, у нас есть записи о самом хосте (❺) – суть файла зоны. Здесь приведены два типа записей о хостах: CNAME и A. Мы уже говорили при рассмотрении программы dig, что CNAME – это ссылка на *каноническое имя*, *псевдоним*. A-запись указывает IP-адрес. В нашем примере видно, что *AbsoluteBSD.com* является псевдонимом для *www.AbsoluteBSD.com*. (Напомним, что в отсутствие явно указанного имени запись по умолчанию относится к домену, который представлен этим файлом!) Хост *www.AbsoluteBSD.com* имеет IP-адрес 209.69.178.30.

Точки, окончание и файлы зон

Вы уже видели (в описании записи SOA, в разделе «Файлы зон»), что при создании зон точки могут подставляться вместо знака @ в адресах электронной почты. Однако точки также выполняют роль символов окончания для имен хостов. При использовании символа @ демон named подразумевает, что все имена хостов являются частью зоны, для которой предназначен этот файл. Нет необходимости каждый раз указывать «www.AbsoluteBSD.com»; named знает, что речь идет об *AbsoluteBSD.com*, поэтому простого имени «www» будет достаточно. (Для каждого хоста есть имя зоны, которое добавляется к его имени.)

Такая система работает хорошо, за исключением случаев, когда хост не является частью данного домена. Например, поскольку серверы имен для *AbsoluteBSD.com* не входят в этот домен, мы не хотим, чтобы они отображались как «blackhelicopters.org.AbsoluteBSD.com».

Здесь на помощь приходит точка. Если поставить точку после имени хоста, named поймет, что задано полное имя хоста, включая доменное имя. В предыдущих примерах видно, что каждое полное имя хоста после записи SOA замыкается точкой. Каждая CNAME-запись, указывающая на *www.AbsoluteBSD.com*, имеет точку; в противном случае она отправила бы нас к «www.AbsoluteBSD.com.AbsoluteBSD.com». Когда вы наберете адрес *www.AbsoluteBSD.com* в веб-броузере, он не сможет найти страницу. Вместо этого придет-

ся набирать «www.AbsoluteBSD.com.AbsoluteBSD.com». Не скажешь, что это удобно, правда?[1]

Активизация изменений

Итак, сервер имен сконфигурирован, а файлы зон настроены. Все выглядит прекрасно. Однако сервер имен не примет изменения, пока вы не предпишете демону named перечитать конфигурационные файлы.

Для принятия изменений используется программа ndc(8) – контроллер демона имен (name daemon controller). ndc может выполнять все функции по управлению демоном named, круг которых различается в зависимости от варианта компиляции named. Для получения полного списка функций запустите `ndc help` как пользователь root:

```
# ndc help
(builtin) start - start the server
(builtin) restart - stop server if any, start a new one
getpid
status
stop
exec
reload [zone] ...
reconfig [-noexpired] (just sees new/gone zones)
dumpdb
stats [clear]
trace [level]
notrace
querylog
qrylog
help
quit
#
```

Для изучения этих параметров следует обратиться к одной из больших книг по DNS. К числу наиболее важных параметров относятся `stop`, `start`, `restart` и `reload`; команда `ndc stop` останавливает сервер имен, `ndc start` запускает его, а `ndc restart` останавливает и затем снова запускает. Повторный запуск сервера имен заставляет его перечитать все файлы зон, чтобы внести поправки в соответствии с измененными данными. Кроме того, его кэш будет очищен от кэшированной информации DNS. (Это нормально, если ваш сервер имен только обслуживает информацию и не предоставляет возможности поиска конечным пользователям.) Для того чтобы named проверял все файлы зон на предмет наличия обновленной информации, но не сбрасывал кэш имен, полученных от чужих серверов имен, применяйте `ndc reload`.

---

[1] На самом деле такое имя хоста фанаты DNS посчитают забавным. Запомните это прежде, чем стать фанатом DNS.

Запуск named во время начальной загрузки

При работе с сервером имен обычно надо автоматически запускать его во время начальной загрузки. Для этого предназначен параметр named_enable файла *rc.conf*:

```
named_enable="YES"
```

Запуск named вручную выполняется при помощи ndc(8) в соответствии с рекомендациями из предыдущего раздела.

Проверка DNS

Создав первую зону, выведите всю информацию о домене и проверьте свою работу. (Ключевое слово axfr программы dig запрашивает список всех хостов в домене.)

```
# dig @primarynameserver domainname axfr
```

Теперь изучите результаты. Все ли имена представлены так, как ожидалось? Есть ли хосты с двойными доменными именами, например «AbsoluteBSD.com.AbsoluteBSD.com»? Если да, вы забыли точку. На месте ли все почтовые серверы и серверы имен? Если нет, внесите исправления.

Для дополнительной проверки можно применить dnswalk(1) (*/usr/ports/net/dnswalk*). Этот инструмент выявляет различные неполадки в конфигурации, однако не способен обнаружить концептуальные проблемы. Если для определенного хоста есть CNAME, но каноническое имя представляет собой значение CNAME, указывающее на первый хост (т. е. получилась петля), dnswalk отловит такую ошибку. Однако если предпочтительный почтовый ретранслятор задан как «mail.whitehouse.gov»,[1] dnswalk это пропустит.

Домен в dnswalk можно указывать так:

```
# dnswalk AbsoluteBSD.com.
```

Ошибки в конфигурации named

Ошибки в конфигурации DNS появляются в */var/log/messages*. В виде сообщений об ошибках появляются и уведомления о запуске, повторном запуске и перезагрузке named. Если сервер имен не обслуживает информацию о домене, проверьте файл протокола. Сообщения, содержащиеся в нем, обычно достаточно ясны; в них представлен номер строки, которая, возможно, послужила причиной ошибки.

Защита named

named – притягательная мишень для хакеров, поскольку он предоставляет много информации о сети и по умолчанию запускается как root. Если кто-

[1] *http://www.whitehouse.gov* – сайт Белого Дома (США). – *Примеч. перев.*

нибудь проникнет в named, он завладеет данной машиной. Рассмотрим обе проблемы по отдельности.

Пример с dig, в котором мы получили весь список хостов в домене, описывает *передачу зоны* (*zone transfer*). Предполагаемый взломщик проявит особый интерес к этой информации, особенно если ваши хосты имеют наглядные имена. («Да-да, компьютер «ceo.AbsoluteBSD.com», наверное, принадлежит президенту компании! Хорошо бы его взломать.»)

Назначение сервера имен состоит в том, чтобы обслуживать имена, поэтому он не может полностью прикрыть доступ взломщиков к машине. Однако named можно настроить так, чтобы он давал ответы только на особые запросы, а не выставлял напоказ свое содержимое. Таким образом, если кто-нибудь выдаст запрос о конкретном хосте, сервер имен ответит. Если же будет запрошен весь список хостов, запрос будет отклонен.

Для того чтобы передачу зоны могли выполнять только определенные хосты, применяйте параметр `allow-transfer`:

```
options {
        directory "/var/named";
        allow-transfer {
                192.168.87.3; 10.115.4.3 ;
                } ;
} ;
```

В данном примере хосты с адресами 192.168.87.3 и 10.115.4.3 – единственные системы, которым позволено выполнять передачу зоны. Замените эти IP-адреса на адреса ваших вторичных серверов имен и своей рабочей станции, и тогда значительная часть информации о вашей сети будет скрыта от посторонних глаз. В список можно добавить и настольные машины, чтобы они также могли выполнять передачу зоны при отладке DNS.

Примечание

Можно определить и более короткий список доступа. Для получения подробной информации обратитесь к */usr/src/contrib/bind/doc*.

А как насчет хакеров, атакующих сам демон named? Можно сделать две вещи. Во-первых, запустить named в клетке (см. главу 8), чтобы удачливый взломщик не смог получить доступ больше ни к чему. Во-вторых, запустить named не от имени root, а от имени другого пользователя. Убедитесь в наличии такой настройки в файле */etc/rc.conf* или */etc/defaults/rc.conf*. (К моменту выхода этой книги такой вариант может быть выбран по умолчанию, поэтому проверьте, так ли это.)

```
named_flags="-u bind -g bind"
```

Если такие ключи установлены, для получения привилегий root злоумышленнику потребуется взломать named, потом проникнуть в раздел root клетки как обычному пользователю, а затем выбраться из тюрьмы в основную систему. Что-нибудь подозрительное вы заметите раньше, чем он проделает все эти действия.

Управление порядком предоставления информации

Порядок, в котором проверяются файл *hosts* и сервер имен, оказывает значительное влияние на работу программы или системы. К примеру, брандмауэрам зачастую нужны особые записи в *hosts*, которые им приходится проверять до обращения к глобальной таблице DNS. Другим хостам такая возможность не нужна. Файл */etc/host.conf* позволяет управлять порядком обращения к источникам информации. В нем могут быть только две записи: `hosts` и `bind`. Каждая запись размещается в отдельной строке. Источники информации о хостах опрашиваются в порядке, представленном в этом файле.

Например, если к файлу *hosts* следует обращаться до запросов к сервису имен, то файл *host.conf* должен выглядеть так:

```
hosts
bind
```

Однако если необходимо, чтобы перед обращением к файлу *hosts* опрашивался сервер имен, можно поступить так:

```
bind
hosts
```

Примечание

Второй источник информации проверяется лишь в случае неудачного обращения к первому. Если записи в */etc/hosts* и DNS противоречат друг другу, приоритет отдается первому источнику.

Дополнительная информация о BIND

По мере расширения сети вам понадобится дополнительная информация о BIND. Хорошим источником является документация в */usr/src/contrib/bind/doc*, но она может оказаться трудной для восприятия. Стандартное руководство по BIND – книга Пола Альбитца (Paul Albitz) и Крикета Ли (Cricket Liu) «DNS and BIND» (O'Reilly).[1] Читать ее легко. Настоятельно рекомендую – это единственная книга, которую я автоматически заказываю при выходе каждого нового издания, даже не бросив на него беглый взгляд.

[1] Пол Альбитц и Крикет Ли «DNS и BIND», 4-е издание. – Пер. с англ. – СПб: Символ-Плюс, 2002 г.

13

Управление малыми сетевыми сервисами

Даже серверу очень узкого, специализированного назначения (например, веб-серверу) необходимо множество меньших, вспомогательных сервисов, таких как основные инструменты администрирования. В этой главе будут рассмотрены малые серверы Интернета, такие как сервер времени, SSH и inetd, а также инструменты, которые FreeBSD предоставляет для их поддержки. Кроме того, будут описаны несколько базовых инструментов, применяемых для управления большими серверами, – например, средства управления пропускной способностью и сертификаты безопасности.

Примечание

В этой главе представлены ссылки на темы, которые не будут охвачены в этой книге. По мере возможности я ссылаюсь на авторитетные источники для получения дополнительной информации. (Если вы запускаете крупномасштабный сервер Интернета, скажем, обрабатывающий миллион и более сообщений электронной почты в час, – вам, возможно, захочется прочесть руководство, содержащее больше сведений, чем несколько страниц этой книги!)

Управление пропускной способностью

Сегодня аппаратные средства компьютеров относительно недороги, а программы дешевы. Однако стоимость полосы пропускания велика. Если ваша компания предлагает клиентам веб-сервис с «неограниченной полосой пропускания», канал быстро заполнится, а надлежащего дохода не будет. Таким образом, может оказаться важным ограничить полосу пропускания, которую может занимать любой сайт, а также полосу пропускания, которую использует любой сервис. Именно здесь на помощь приходит dummynet. Луиджи Риццо (Luigi Rizzo) изобрел dummynet для эмуляции плохих и обрывающихся линий связи. Такой механизм позволяет тестировать сетевые протоколы в неблагоприятных обстоятельствах. dummynet довольно гибок – на веб-странице Луиджи Риццо (*http://www.iet.unipi.it/~luigi/ip_dummynet/*) можно найти пример эмуляции линии ADSL к Луне! (dummynet является частью пакета IPFW, который затрагивался в главе 8.)

Хотя инструмент dummynet разработан для тестирования сетевых протоколов, он также используется в целях регулировки ширины полосы пропускания, занимаемой тем или иным сетевым сервисом. Управление полосой пропускания – лишь один из побочных результатов такого экспериментирования. Кроме того, поскольку dummynet работает на определенных портах, IP-адресах и протоколах, его можно использовать для ограничения полосы пропускания туннелей IPSec, программы sendmail и т. п.

Для использования dummynet ядро должно быть скомпилировано с поддержкой IPFW. Если вы следовали примеру главы 4, все должно быть в порядке. Однако для дополнительной проверки запустите `kldstat -v | grep ipfw`. В результате будет выдан список всех модулей IPFW. Если вы обнаружите, что поддержка IPFW в ядре отсутствует, добавьте следующие строки в конфигурацию ядра, затем пересоберите его и перезапустите систему.

```
options         IPFIREWALL
options         IPFIREWALL_VERBOSE
options         DUMMYNET
options         IPFIREWALL_DEFAULT_TO_ACCEPT
```

Примечание

Поскольку мы применяем IPFW для управления полосой пропускания, а не для фильтрации пакетов, зададим режим принятия по умолчанию. Если вместо IPF для фильтрации пакетов используется IPFW, «принятие по умолчанию» исключите полностью.

Конфигурирование IPFW

В IPFW фильтрация пакетов организована так: каждый пакет сравнивается с каждым правилом, по порядку. Правила определяют, будет ли этот пакет принят, отброшен или передан другой программе, например divert(4) или dummynet.

Поскольку мы применяем IPFilter для фильтрации пакетов, нам следует позаботиться только о подмножестве правил IPFW, которые обеспечивают ограничение трафика. В этом подмножестве программе dummynet необходимы два правила: правило IPFW для перенаправления пакета программе dummynet и правило dummynet, описывающее разрешенный трафик. Вкратце рассмотрим примеры этих правил.

Для конфигурирования IPFW применим `ipfw(8)`, войдя в систему как root. Однако поскольку `ipfw` (подобно многим другим программам) выполняется по-разному в зависимости от заданных аргументов, сначала проверим исходные правила с помощью `ipfw list`.

```
# ipfw list
65535 allow ip from any to any
#
```

Как видно из предыдущего примера, правило начинается с номера, затем следует имя правила. Правила IPFW нумеруются от 1 до 65 535. Кажется, довольно просто. Поскольку мы использовали параметр настройки ядра «разрешать по умолчанию» (default to accept), последнее возможное правило (с номером 65 535) пропускает весь трафик. Не поступи мы так, последнее возможное правило запретило бы весь трафик.

Для того чтобы IPFW пересылал пакеты программе dummynet, необходимо создать правило IPFW для перенаправления трафика определенного типа правилу dummynet. Правило IPFW-dummynet должно содержать следующее:

- Номер правила IPFW
- Оператор, согласно которому трафик перенаправляется правилу другого типа (правило dummynet)
- Номер другого правила
- Описание трафика

```
number pipe pipenumber ip from sourceaddr sourceport to destaddr destport
```

В предыдущем примере `number` – это номер правила IPFW, а `pipenumber` – номер канала (pipe), обслуживающего это правило. (*pipe* – это дополнительное правило IPFW, выполняющее специальную обработку, такое как dummynet.) Поля `sourceaddr` и `sourceport` определяют IP-адрес и номер порта источника трафика, а `destaddr` и `destport` задают место назначения. Номера портов можно не указывать, и тогда затрагивается весь входящий и исходящий трафик данного IP-адреса. (В полях источника и места назначения можно применять ключевое слово `any` для указания всех возможных адресов.)

Вот простое правило IPFW-dummynet:

```
100 pipe 1 ip from 192.168.99.100 80 to any
```

В этом примере число 100 – номер правила IPFW, а `pipe` – это маркер. Он указывает, что данное правило должно перенаправлять трафик другому набору правил. Номер правила pipe – 1. Оставшаяся часть правила представляет собой описание трафика.

Описание трафика

Описание трафика, который необходимо «прокачать» через dummynet, очень важно. Ошибочное описание трафика способно излишне сузить или расширить полосу пропускания.

Вот основной формат описания трафика:

```
protocol from address port to address port
```

Для Интернета параметр `protocol` почти всегда имеет значение `ip`. Поля `from` и `to` определяют, откуда и куда идет трафик. Поля `address` – это IP-адреса, а `port` – номера портов. Все возможные IP-адреса и порты можно указать с помощью ключевого слова `any`.

Например, пусть IP-адрес нашего веб-сервера 192.168.99.100. Опишем весь трафик, исходящий с веб-сервера и направляющийся по любому адресу в Интернете. Описание данного трафика может быть таким:

```
ip from 192.168.99.100 80 to any
```

Создание правил IPFW

Пусть требуется фильтровать трафик веб-сервера с адресом 192.168.99.100, который слушает порт 80. Описание этого трафика приведено в предыдущем разделе. Теперь добавим к нему информацию, необходимую для перенаправления трафика правилу dummynet.

Для создания правила IPFW нужны номер правила IPFW и номер правила pipe. Правила IPFW анализируются в порядке нумерации, однако схема нумерации может быть любой. Поскольку мы не применяем IPFW для фильтрации пакетов, а только перенаправляем пакеты программе dummynet, порядок не так важен. Обычно я нумерую правила с приращением 100, чтобы оставить место для добавления новых правил к существующим. Порядок в правилах pipe не важен, поэтому я нумерую их последовательно. Итак, правило IPFW получает номер 100, а правило pipe – 1.

Правило IPFW будет выглядеть так:

```
100 pipe 1 ip from 192.168.99.100 80 to any
```

Добавление правил IPFW

Содержимое правила IPFW определено, и его надо добавить в список правил IPFW. Для этого выполняется команда `ipfw add`:

```
# ipfw add 100 pipe 1 ip from 192.168.99.100 80 to any
```

Это правило предписывает IPFW принимать любой трафик из порта 80 на 192.168.99.100 и перенаправлять его через правило pipe 1.

Создание правил pipe

Итак, IPFW направляет трафик определенного описания к правилу dummynet (или pipe). Это помогло бы, если бы правило существовало. Синтаксис правил dummynet таков:

```
pipe pipenumber config bw bandwidth
```

Поле `pipe` в начале предыдущего оператора указывает на то, что это правило pipe. Параметру `pipenumber` присваиваем значение 1 – это тот же номер, что и для правила IPFW. В поле `bandwidth` определим допустимую пропускную способность соединения. В нашем примере она составит 128 Кбит/с.

Добавим это правило в список IPFW с помощью `ipfw add`:

```
ipfw add pipe 1 config bw 128Kbit/s
```

Итак, теперь весь исходящий трафик веб-сайта с указанным IP-адресом будет перенаправляться правилу dummynet, которое ограничит скорость передачи пакетов до 128 Кбит/с.

Просмотр правил IPFW

Для просмотра правил IPFW запустите `ipfw list`:

```
# ipfw list
00100 pipe 1 ip from 192.168.99.100 80 to any
65535 allow ip from any to any
#
```

Как видно из листинга, правило IPFW перенаправляет трафик правилу dummynet. Однако правило dummynet здесь не показано. Правила pipe хранятся в отдельном списке.

Для просмотра правил pipe запустите `ipfw pipe list`:

```
# ipfw pipe list
❶ 00001: ❷ 128.000 Kbit/s    0 ms   50 sl. 1 queues (1 buckets) droptail
    mask: 0x00 0x00000000/0x0000 -> 0x00000000/0x0000
BKT Prot ___Source IP/port____ ____Dest. IP/port____ Tot_pkt/bytes Pkt/Byte Drp
  0 tcp   ❸ 192.168.99.100/80   ❹ 163.62.168.2/2415  128050681 35518324182  0   0 50486587
#
```

Примечание

Ширина вывода команды `ipfw pipe list` намного больше 80 символов. Если возможно, используйте эмулятор терминала и сделайте свое окно как можно шире.

Четыре строки, представленные в предыдущем листинге, описывают как правило dummynet, так и связанное с ним правило IPFW. Большую часть этих подробностей понимать необязательно – они достались нам в наследство от dummynet как инструмента, предназначенного для имитации проблем с трафиком. Первое поле (❶) – номер правила dummynet. За ним следует правило, в соответствии с которым dummynet разрешает трафик (❷). Следующее интересное поле (❸) – IP-адрес источника – в данном случае нашего веб-сервера. В момент выполнения команды один хост с определенным IP-адресом (❹) получает трафик от нашего веб-сервера.

Очереди dummynet

dummynet работает так: пакеты помещаются в очередь, а затем обрабатываются по порядку. Перекрытие сайта с интенсивным трафиком может привести к заполнению очереди, следовательно, если веб-сайт время от времени блокируется на несколько секунд после реализации dummynet, то определенный для очереди пакетов размер может может оказаться недостаточным.

Для устранения проблемы измените правило pipe: добавьте размер очереди и увеличьте его значение до максимально возможных 1000 Кбайт:

```
ipfw add pipe 1 config bw 128Kbit/s queue 1000Kbytes
```

Однако чем больше размер очереди, тем больше потребление памяти ядра. Не переборщите.

Ограничение трафика по направлению

Следует иметь в виду еще одно обстоятельство: нельзя перекрывать входящий трафик. Если разместить пиратскую копию фильма «Звездные войны» на своем веб-сервере, то количество входящих запросов в секунду превысит все мыслимые пределы. Никакое конфигурирование не заставит 30 миллионов пользователей отказаться от искушения щелкнуть по ссылке, ведущей на страницы вашего сервера. Лучшее, что можно предпринять, – ограничить ответы на такие запросы. Другими словами, надо лишь перекрывать исходящие соединения. В большинстве случаев такая схема годится, – в конце концов, именно вы обслуживаете веб-страницы или посылаете почту!

Если для управления входящим трафиком применяется правило dummynet, то скорость передачи данных снизится, а никакого влияния на входящий трафик оказано не будет. В результате вы создадите очереди соединений и лишь причините вред самому себе. Если канал «переполнен» трафиком, надо либо полностью отклонять трафик такого вида (см. главу 8), находить запрашиваемое содержимое и удалять его, либо обратиться за помощью к провайдеру.

Шифрование открытым ключом

Многие функции обеспечения безопасности, реализованные в серверных демонах, опираются на шифрование открытым ключом. Оно обеспечивает конфиденциальность, целостность и достоверность данных. Различные сервисы Интернета также используют такое шифрование. Знание его основ поможет организовать защищенные веб-страницы (https) и защищенную почту POP3 (pop3ssl).

Примечание

Тот, кто знаком с шифрованием открытым ключом, вероятно, может пропустить этот раздел.

В системах шифрования ключ применяется для преобразования удобочитаемых сообщений с открытым текстом (cleartext) в версию с зашифрованным текстом (ciphertext), а также для обратного преобразования. Хотя в терминах «открытый текст» и «зашифрованный текст» есть слово «текст», сообщения не ограничены только текстом. В них могут быть графические, исполняемые файлы и любые другие пересылаемые данные. У каждой криптографической системы есть три основных назначения: поддерживать целостность и конфиденциальность данных и обеспечивать невозможность отказа от авторства. *Целостность* означает, что сообщение не будет искажено. *Конфиденциальность* подразумевает, что сообщение сможет прочесть только предопределенный круг лиц. *Невозможность отказа от авторства* означает, что автору не удастся впоследствии заявить, будто он не писал того или иного сообщения.

Старые системы шифрования предполагали наличие единого ключа для шифрования и дешифрования сообщений. (Для преобразования сообщения требовалось выполнить много операций, как в случае с системой Enigma,[1] применявшейся коалицией союзников во время второй мировой войны. Как бы то ни было, единый ключ сделал шифрование возможным.) Типичным примером является любой код, для которого необходим ключ или пароль. Планшеты для создания разовых сообщений, которые стали известны благодаря фильмам про шпионов, служат примером первых систем шифрования одним ключом.

В отличие от систем шифрования одним ключом, в системах шифрования открытым ключом (несимметричное шифрование) применяются два ключа: секретный (private) и открытый (public). Сообщения шифруются одним ключом, а расшифровываются другим. (Математическое объяснение довольно сложно, чтобы приводить его здесь, однако такая схема работоспособна. Система основана на свойствах очень больших чисел.) Владелец ключа держит секретный ключ в тайне, а открытый ключ раздает направо и налево. Владелец ключа применяет секретный ключ, а все остальные – открытый. Владелец ключа может шифровать сообщения, которые может открыть любой из указанных им субъектов. Однако сообщения, зашифрованные открытым ключом, может прочесть только владелец ключа.

Криптография с открытым ключом прекрасно удовлетворяет наши потребности в целостности, конфиденциальности и невозможности отказа от авторства. Если автор хочет, чтобы его сообщение смог прочесть любой желающий, но никто не смог его исказить, он зашифровывает сообщение своим секретным ключом. Все, у кого есть его открытый ключ, смогут дешифровать и прочесть это сообщение. (Искажение зашифрованного сообщения сделает его нечитаемым.)

Такое шифрование сообщений гарантирует, что у автора есть секретный ключ. Если автор хочет послать сообщение, которое разрешено прочесть только указанному получателю, он сможет зашифровать текст открытым ключом получателя. И только получатель, у которого есть соответствующий секретный ключ, сможет прочесть это сообщение.

Описанная система работает отлично, пока секретный ключ остается секретным. Если он похищен, потерян или раскрыт, то становится бесполезным. Неосторожный субъект, у которого похитили ключ, может даже столкнуться с тем, что кто-то другой подписывает документы его именем. Внимательно относитесь к ключам. Иначе кто-нибудь, воспользовавшись вашим сертификатом, сделает срочный заказ на профессиональные графические станции стоимостью в полмиллиона долларов и укажет адрес заброшенного дома на другом конце страны![2]

---

[1] Описание роторных криптографических машин Enigma можно найти по адресу *http://crypto-r.narod.ru/glava2/glava2_4.html*. – *Примеч. науч. ред.*

[2] Это чистая правда. Берегите свои секретные ключи!

Примечание

«FreeBSD. Подробное руководство» не является всеохватывающим руководством по криптографии. Многие концепции представлены здесь обобщенно. Если вам действительно интересна криптография, обратитесь к книге Брюса Шнейера (Bruce Schneier) «Applied Cryptography» (Прикладная криптография), John Wiley & Sons. Запаситесь калькулятором и дополнительным мозгом (пригодится, когда будет переполнен ваш собственный).

Сертификаты

Интересная особенность шифрования открытым ключом заключается в том, что автор и получатели сообщений необязательно должны быть людьми – это могут быть программы. Шифрование открытым ключом реализовано в протоколе защищенных сокетов (Secure Sockets Layer, SSL) сервера Apache и в защищенном сервисе POP3 на базе SSL. Такое шифрование применяют и другие программы. Криптография с открытым ключом – это важный компонент *подписанных сертификатов* (*signed sertificates*), используемых защищенными веб-сайтами. Покупая что-нибудь с помощью Netscape в режиме онлайн, возможно, вы и не догадываетесь, что броузер неистово зашифровывает и расшифровывает веб-страницы. Вот почему ваш компьютер может пожаловаться на «недействительные сертификаты» (invalid certificates) или на то, что истек срок действия открытого ключа либо ключ был поврежден. (Более подробно об использовании сертификатов рассказывается в главах 14 и 15.)

Многие компании, например VeriSign, предоставляют услуги по заверению открытых ключей. Такие компании называются Certificate Authorities (CAs, источники сертификатов). Другие компании, которым необходимо заверить сертификаты, предоставляют доказательства подлинности предоставляемой о себе информации (например, корпоративные документы и деловые книги), а Certificate Authority применяет свой сертификат для заверения сертификатов этих компаний. Заверяя сертификат, Certificate Authority говорит: «Я проверил документы этой персоны (организации). Он (или она) является тем, за кого себя выдает». Однако Certificate Authority не гарантирует, что эта персона не применит сертификат для веб-сервера, продающего поддельные или опасные продукты, или для шифрования требования выкупа. Подписанный (заверенный) сертификат гарантирует определенную техническую безопасность, а не персональную честность или одностороннюю техническую безопасность. Но если кто-нибудь взломает сервер, то неприятности у вас все-таки могут быть.

В веб-броузерах и других программах, применяющих сертификаты, есть сертификаты основных CAs. Получая сертификат, подписанный Certificate Authority, броузер принимает его. По сути, он говорит: «Я доверяю Certificate Authority, а Certificate Authority доверяет этой компании; следовательно, я буду доверять этой компании.» Пока вы доверяете источнику сертификатов, схема работает.

Создаем запрос

Чтобы получить сертификат для защиты серверной программы, надо подготовить соответствующий запрос. Далее запрос сертификата направляется центральному Certificate Authority. Сам запрос довольно прост. Хотя команда длинная, достаточно ответить на несколько вопросов. (Поскольку эти команды используются в книге только один раз, они не будут подробно рассмотрены; если вы интересуетесь дополнительной информацией, обратитесь к openssl(1).)

Примечание

Запрос сертификата должен рассматриваться как секретный, поскольку хакер может воспользоваться им для входа в вашу сеть. Убедитесь, что этот файл может читать только root!

Рассмотрим запрос сертификата. Введите дословно такую команду:

```
# openssl req -new -nodes -out req.pem -keyout cert.pem
```

В ответ должно появиться следующее:

```
Using configuration from /etc/ssl/openssl.cnf
Generating a 1024 bit RSA private key
.................++++++
...++++++
writing new private key to 'cert.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) [AU]:US
State or Province Name (full name) [Some-State]:MI

(Применяется конфигурация, представленная в /etc/ssl/openssl.cnf
Создание 1024-битного секретного ключа RSA
.................++++++
...++++++
Запись нового секретного ключа в 'cert.pem'
-----
Сейчас вам будет предложено ввести информацию, которая будет включена в ваш запрос
сертификата.
Вводимые сведения называются Отличительным именем (Distinguished Name), или DN.
Здесь довольно много полей, но часть из них можно оставить пустыми.
Для некоторых полей будет указано значение по умолчанию.
Если вы введете  '.', поле останется пустым.
-----
Название страны (двухбуквенный код) [AU]:US
Название штата или провинции (полное название) [Some-State]:MI)
```

Введите двухбуквенный код для страны и штата или провинции, где вы проживаете (в нашем примере соответственно US и MI; показано жирным шрифтом). Если вы не знаете двухбуквенные коды, спросите кого-нибудь из тех, кто временами покидает серверную комнату. (Кроме того, коды описаны в стандарте ISO 3166 – значит, их можно быстро найти в Сети.)

```
Locality Name (eg, city) []:Detroit      (Название местности)
```

Простого названия города для указания местности будет достаточно. Если вы работаете в филиале компании, можно указать город, в котором располагается ее головной офис.

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:BlackHelicopters
Foundation
(Название организации (например, компании))
Organizational Unit Name (eg, section) []:Network Support
(Название подразделения компании (например, отдела))
```

В вышеприведенных строках запрашивалось название компании и отдела, в котором вы работаете. Если такой компании нет (у меня нет), придется что-нибудь придумать.

```
Common Name (eg, YOUR name) []:magpire.blackhelicopters.org
(Принятое имя (например, ВАШЕ имя))
```

Последняя строка оказывается камнем преткновения для большинства администраторов. «ВАШЕ» означает имя сервера, а не имя администратора. Если не указать здесь имя сервера, запрос будет бесполезен.

```
Email Address []:mwlucas@blackhelicopters.org      (Адрес электронной почты)
```

Поскольку это личный сертификат для моего собственного веб-сервера, я не обращаю внимания на адрес электронной почты. Если данный запрос исходит от компании, укажите здесь общий корпоративный адрес, например *webmaster@AbsoluteBSD.com*.

```
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:RodentsRule
(Пожалуйста, введите 'дополнительные атрибуты', которые будут
посланы вместе с запросом сертификата
Запрашиваемый пароль []:RodentsRule)
```

Запрашиваемый пароль также называют идентификационной *фразой* (*passphrase*). Повторюсь: он должен храниться в тайне, поскольку тот, кому станет известна ваша идентификационная фраза, сможет выдавать себя за вас! Представленная парольная фраза не самая лучшая; в ней отстутствуют символы, не относящиеся к алфавитно-цифровым, такие как тире, запятая или восклицательный знак. В ней даже нет цифр. В реальных запросах сертификатов (и везде в вашей сети) пароли должны быть более сложными.

```
An optional company name []:      (Необязательное название компании)
```

Уверен, к этому времени вы уже ввели достаточно вариантов названия компании, поэтому просто нажмите Enter. После этого найдите файл *req.pem* в текущем каталоге. Он должен выглядеть примерно так:


```
-----BEGIN CERTIFICATE REQUEST-----
MIICIDCCAYkCAQAwgcAxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJNSTEQMA4GA1UE
BxMHRGV0cm9pdDEkMCIGA1UEChMbQmxhY2tIZWxpY29wdGVycyBGb3VuZGF0aW9u
MRgwFgYDVQQLEw9OZXR3b3JrIFN1cHBvcnQxJTAjBgNVBAMTHG1hZ3BpcmUuYmxh
Y2toZWxpY29wdGVycy5vcmcxKzApBgkqhkiG9w0BCQEWHG13bHVjYXNAYmxhY2to
ZWxpY29wdGVycy5vcmcwgZ8wDQYJKoZIhvcNAQEBBQADgYOAMIGJAoGBANCjXf0h
WX/nlKb5Sc9m7Nofvc3Nck5j7XzNnd5OUIc93Jj+Egw/Knlrniptp Nicvqz QJ6zs
7jOk1uMUMbHfllxUOUtRGfLthCvfstB4OZzdMYUAfAT1r15i7fnaCRagshekelOh
deadbeefTCk6mC7OYcsGuqrVuQkEcA/kPDxdAgMBAAGgHzAdBgkqhkiG9w0BCQcx
EBMOR2VyYmlscOFyZUNvb2wwDQYJKoZIhvcNAQEEBQADgYEAwC7lNqZbHFKaOjiw
h35gU6TAC8NEODRLuEulLWClEIPsTK6HHV7KU4uOq42HEunf61dpPaPkGO3htoeu
yOc5Rjk9F11cvRbBjpajv+T1lxTBGveuhatsn43d9Epi3glrcpueisd87LMxtnht
OBf9nz6GaH+2o2BsGxwH3yws5oO=
-----END CERTIFICATE REQUEST-----
```


Также будет создан файл *cert.pem*:


```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQDQo139IVl/55Sm+UnPZuzaH73NzXJOY+18zZ3edFCHPdyY/hIM
Pyp5a54qbaTYnL6sOCes7O4zpNbjFDGx35ZcVNFLURny7YQr37LQeNGc3TGFAHwE
9a9eYu352gkdSbY5YlPr+7K63bRkskwpOpguzmHLBrqq1bkJBHAP5Dw8XQIDAQAB
AoGAO8olXC4bdOELo5IbCdmoFJY2EW1HzZkrbLGMBTz1+tvKhPmCeIn9hRBHIkeL
jxvUNLfuNssrNBeQEUEvQJcfgk+QW8zq5UV6xin7Rb1JYu+1TzyBt1QMAx99cDEq
WWOoqvYIz1IzQq6FA5/J93Kj3yJ7I6NOCs8c9BxYvnjd6WECQQDOARUKZhwLD7gQ
HM3aIMXV7hOnzqj1Ygz2Rw/GEj+eWiam9NDlxIjqCuXAp34rDcyp++ZFX8flOJQ+
yHOt7625AkEA2uUvUhobOvTAFBofrFHigRQRD8YFDbXIPLtrXxqAmuD1SyABBgBy
yGpsmXwdBP/lxR1xu4n+Mu2KVPiNZpZ1xQJASlNGEHvYEPqBy86qWcZf3PGCSgzm
ZJCweBhfUqteW6MEYRjzxPmf5wLYx119zimO7TyBASLS5hzc817l9daraQJBAJ6B
8YdRcq6LHwAvfpoI3aO8u7IhYY1xAiPAT9sZVOFSXy3cagFPl867ChMGxfjV2Suo
y6/TGCkGy/IF3lbYQOUCQGABvzCfcw3/xVY7co6k8kSu1Mf1dj/MYZhOoI7qrbUN
O+Cez+e2UvoiahCW3IWlmBFBZ8HJUoGzkCO+wVmZzZO=
-----END RSA PRIVATE KEY-----
```


Если таких файлов нет или они выглядят совсем по-другому, значит, программа openssl была выполнена неверно. Вернитесь назад и попробуйте снова либо пошлите *req.pem* вашему Certificate Authority. В ответ вы должны получить файл, который будет похож на один из вышеприведенных. Сохраните эту информацию в файле с именем *signature.pem* и выполните такую команду:

```
# cat signature.pem >> cert.pem
```

В результате содержимое *signature.pem* будет добавлено в конец сертификата. Таким образом, будет создан полный подписанный сертификат. Он подходит для всех сервисов хоста, и его можно применять для веб-страницы, соединения pop3ssl и в любых других приложениях, где нужен сертификат.

Сам себе CA

На первых порах, возможно, хлопотное обращение к Certificate Authority для подписания каждого сертификата вас не привлечет. Наверняка вам нужны несколько сертификатов для учебных целей. Подписание сертификата – это простая математическая операция, которую можно прекрасно выполнить самому.

Примечание

Если вы подписываете собственные сертификаты, клиентские программы будут выдавать предупреждения о том, что «сторона, подписавшая сертификат, неизвестна» (certificate signer is unknown). Ничего удивительного, в конце концов, люди за стенами моего офиса понятия не имеют, кто такой Майкл Лукас и почему он подписывает сертификаты веб-сайта! Компании VeriSign и другим CAs оказывается доверие. Мне доверяют люди, которые меня знают,[1] но мне не доверят подтверждать подлинность сведений о других людях.

Для подписания своих сертификатов сначала создайте каталог (право на его чтение должно быть только у root), а затем выполните все необходимые операции, относящиеся к CA:

```
# mkdir ca
# chown root.wheel ca
# chmod 700 ca
```

Далее, для создания ключа источника сертификата выполните следующую команду openssl:

```
# openssl genrsa -des3 -out ca.key 1024
```

Введите запрашиваемую идентификационную фразу и убедитесь, что запомнили ее, иначе созданный ключ будет бесполезен. (Сертификат нельзя использовать без идентификационной фразы!) Наконец, создайте сертификат для своего CA с помощью только что созданного ключа:

```
# openssl req -new -x509 -days 365 -key ca.key -out ca.crt
```

Получив приглашение, введите идентификационную фразу (если вы ее уже забыли, остается только создать новый ключ), а затем ответы на вопросы, идентичные вопросам, задаваемым при подготовке запроса сертификата. Теперь у вас есть CA-ключ (`ca.key`) и CA-сертификат (`ca.crt`), которые можно применить для подписания запроса, созданного ранее. Приведенная команда длинна, но она никогда не меняется, поэтому подробно она рассматриваться не будет. Поверьте на слово.

Далее снова последует запрос идентификационной фразы. После того как она введена, сам процесс подписания проходит очень быстро. В результате будет создан файл *signature.pem* – файл-сигнатура, который CA послал бы

[1] Во всяком случае большинство из них. Многие из них. По крайней мере какие-то из них. Да бог с ними!

вам в ответ на запрос сертификата. Добавьте его к вашему открытому ключу (как обсуждалось в предыдущей сессии), и вы получите полный сертификат!

Примечание

Сертификат, подписанный самостоятельно, не следует применять в системе, где пользователи его увидят, поскольку выдаваемые предупреждения смутят их или раздосадуют. Или даже отпугнут. Потратьте сотню долларов или около того и получите настоящую CA-подпись своих рабочих сертификатов. (Конечно, если вы применяете эти сертификаты только для внутренних нужд, можно загрузить ваш корпоративный сертификат и установить его в веб-броузерах. В Netscape 6 такая возможность предоставляется в меню Edit → Preferences → Advanced → Privacy & Security → Certificates. В различных версиях Internet Explorer эта функция расположена в других местах.)

SSH

Одна из самых мощных возможностей UNIX – удаленное администрирование. Неважно, где находится сервер – перед вами или же в подвале запертой лаборатории посреди совершенно секретного военного оборудования, охраняемого свирепыми собаками и бешеными горностаями. Если к нему есть доступ по сети, имя пользователя и пароль известны, то им можно управлять.

Многие годы telnet(1) был стандартным средством для доступа к удаленному серверу. telnet хорош. С его помощью можно соединиться с произвольным TCP-портом хоста и непосредственно «разговаривать» с серверами по сети. (Далее в этой главе такая возможность будет применена для тестирования различных сервисов.) Однако telnet, как протоколу удаленного администрирования, сопутствует одна серьезнейшая проблема: в большинстве его реализаций все данные, посылаемые по этому протоколу, передаются в нешифрованном виде. Любой владелец анализатора пакетов (packet sniffer), прослушивающий ваше соединение, может заполучить ваш пароль. От анализатора пакетов не сможет защитить даже самая лучшая схема выбора паролей в мире. Я встречал анализаторы в магистралях Интернета и в маленьких локальных сетях. Единственная защита от них – управлять соединениями так, чтобы злоумышленники не смогли извлечь полезную информацию.

Именно здесь на помощь приходит SSH, или secure shell. В значительной степени SSH работает так же, как и telnet: он предоставляет отлично конфигурируемое терминальное окно на удаленном хосте. Однако, в отличие от telnet, все данные, посылаемые по этому протоколу, шифруются. SSH обеспечивает на только защиту паролей от анализатора, но и шифрование набираемых команд и их вывода. Хотя у telnet есть несколько преимуществ над SSH (он требует меньше процессорного времени и его проще конфигурировать), его преимущества нивелируются преимуществами SSH в обеспечении безопасности.

Если вы ищете дополнительную информацию по этой теме, книга Дэниэла Баррета (Daniel Barrett) и Ричарда Силвермана (Richard Silverman) «SSH, The Secure Shell» (O'Reilly), возможно, является на сегодня лучшей книгой о протоколе SSH.

Тестирование SSH

В отличие от других протоколов, рассматриваемых в этой книге, SSH трудно тестировать вручную. Можно лишь убедиться, что демон SSH запущен. Для этого надо с помощью telnet подключиться к TCP-порту, на котором предположительно работает SSH:

```
# telnet localhost 22
Trying ::1...
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
SSH-1.99-OpenSSH_2.3.0 FreeBSD localisations 20010713
```

Последняя строка вывода сообщает, что SSH запущен и принимает соединения.

Теперь, если только вы не способны шифровать пакеты вручную, на лету, именно этим мы и займемся. Наберите управляющий символ (Control-]) для закрытия соединения. Последует возврат к локальному приглашению на ввод команды.

Включение SSH

Если система еще не сконфигурирована для включения SSH во время начальной загрузки, просто добавьте в */etc/rc.conf* такую запись:

```
sshd_enable="YES"
```

Во время следующей перезагрузки SSH будет включен. Если вы не хотите перегружать систему сейчас, для запуска SSH можно просто набрать sshd, войдя в систему как root.

Основы SSH

В SSH применяется криптография с открытым ключом. Демон SSH предоставляет клиентам открытый ключ, а сам хранит секретный. Каждый блок данных, посылаемый по линии связи, обрабатывается как сообщение, которое ваша локальная система зашифровала открытым ключом; далее сервер расшифровывает данные секретным ключом. Для такой транзакции необходимы как открытый, так и секретный ключи, поэтому безопасность данных не пострадает. Даже если кто-либо перехватит ваш SSH-трафик, он увидит только «мусор».

Создание ключей

Если в системе нет файла */etc/ssh/ssh_host_key* или */etc/ssh/ssh_host_dsa_key*, их можно создать так:

```
# /usr/bin/ssh-keygen -N "" -f /etc/ssh/ssh_host_key
# /usr/bin/ssh-keygen -d -N "" -f /etc/ssh/ssh_host_dsa_key
```

Примечание

Протоколу SSH несколько лет от роду, и это начинает сказываться. Несмотря на то что он является защищенным, администраторам нужна дополнительная гибкость. По этой причи-

не стандарт Secure Shell 2 находит все большее применение. Если иное не оговорено, все приводимые сведения относятся к обеим версиям SSH. Если та или иная функция реализована только в SSH1 или SSH2, это будет отмечено. Некоторые файлы различаются в зависимости от версии SSH – на это также будет указано.

Подтверждение подлинности в SSH

Шифрование открытым ключом сводится на нет, если вы получаете неверный открытый ключ для того или иного хоста. Это случается либо вследствие ошибки пользователя, либо по чьему-то злому умыслу. Самый надежный способ идентифицировать хост состоит в том, чтобы сравнить открытый ключ на сервере с открытым ключом, доступным в сети. Для SSH версии 1 открытый ключ по умолчанию находится в */etc/ssh/ssh_host_key.pub*, а для SSH версии 2 – в */etc/ssh/ssh_host_dsa_key.pub*. Две версии SSH предъявляют различные требования к протоколу, поэтому им требуются разные ключи.

Вы можете скопировать открытые ключи SSH версий 1 и 2 на каждый хост, к которому хотите подключаться, и вручную сравнивать ключи перед подключением, однако ключи могут состоять из сотен символов. Однако сложность не в самом сравнении ключей, а в том, чтобы не выполнять такую проверку. К счастью, SSH позволяет генерировать отпечаток *ключа* (*key fingerpint*), который намного короче самого ключа. Отпечаток нельзя применять для шифрования трафика или установления соединений, однако вероятность того, что два несвязанных между собой ключа имеют одинаковый отпечаток, мала. Для генерации ключа SSH версии 1 введите такую команду:

```
# ssh-keygen -lf /etc/ssh/ssh_host_key.pub
1024 7c:07:0f:1e:74:1a:42:11:b9:08:41:e4:f3:c9:05:a7
root@petulance.blackhelicopters.org
```

Ответом команды будет отпечаток ключа. Первое число, 1024, сообщает количество битов в ключе (1024 – стандартное значение на сегодня). Шестнадцатеричная строка, начинающаяся символами «7c» и заканчивающаяся символами «a7», представляет собой отпечаток открытого ключа. Отпечаток ключа с исходного сервера надо скопировать в место, к которому возможен доступ с клиентских программ – либо на веб-странице, либо в списке. Отпечаток потребуется при установлении соединения с сервером.

Для получения отпечатка ключа SSH2 можно применить ту же команду, если в командной строке заменить имя файла, содержащего ключ SSHv2.

Примечание

Если сервер предлагает SSH1 и SSH2, как это делает FreeBSD по умолчанию, полезно подготовить отпечатки для того и другого открытого ключа. Нет способа определить, какую версию будет применять пользователь, подключающийся к серверу.

Клиенты SSH

Основная трудность при работе с SSH состоит в том, чтобы найти клиентскую программу, которая будет работать на вашей настольной системе. Если

в качестве настольной системы применяется BSD, SSH уже есть в ее составе. В других операционных системах UNIX пакеты SSH обычно тоже доступны. Если это возможно, применяйте OpenSSH (*http://www.openssh.com*). Эта программа, разработанная командой OpenBSD, быстро стала самой распространенной реализацией SSH.

Тем, кто работает с операционной системой Microsoft, я рекомендую MindTerm (хотя у меня также были веские аргументы в пользу Putty и Teraterm). Пакет MindTerm бесплатен при некоммерческом использовании, поддерживает SSH1 и SSH2. Он написан на Java и, следовательно, будет работать на любой платформе, на которой реализована виртуальная машина Java (JVM). (JVM есть в большинстве броузеров.) Документация MindTerm позволит запустить клиент SSH в течение нескольких минут. Поиск в Сети быстро приведет к любой из этих программ, каждая из которых наверняка вам подойдет.

Соединение через SSH

Для подключения к другому хосту с помощью ssh-клиента FreeBSD наберите `ssh hostname`. В ответ должно появиться следующее:

```
# ssh moneysink.blackhelicopters.org
The authenticity of host 'moneysink' can't be established.
RSA key fingerprint is 7c:07:0f:1e:74:1a:42:11:b9:08:41:e4:f3:c9:05:a7.
Are you sure you want to continue connecting (yes/no)? yes
(Идентичность хоста 'moneysink' не может быть установлена.
Отпечаток ключа RSA - 7c:07:0f:1e:74:1a:42:11:b9:08:41:e4:f3:c9:05:a7
Уверены ли вы, что хотите продолжить процесс соединения (да/нет)? Да)
```

Клиент непосредственно выполняет две операции. Во-первых, он извлекает открытый ключ с удаленного хоста. Во-вторых, он сверяет ключ этого хоста с ключами из своего списка SSH. Если ключ хоста есть в списке клиента, а ключ, извлеченный с хоста, соответствует ему, клиент предполагает, что соединение устанавливается с надлежащим хостом. Если ключа сервера нет в клиентском списке хостов, он предъявляет клиенту отпечаток ключа для одобрения.

Изучив отпечаток, необходимо решить, устанавливать соединение или отказаться от него; ваш отпечаток должен быть идентичен отпечатку, созданному на удаленном хосте. Если отпечатки не совпадают, то либо вы подключаетесь не к тому хосту, либо имеете отпечаток для другой версии SSH. Сравните созданный отпечаток с отпечатком на удаленном хосте. Если они совпадают, это требуемый хост. Как только вы признаете ключ, он будет сохранен в файле */.ssh/known-hosts* (для SSH1) или в файле *~/.ssh/known-hosts2* (для SSH2).

Не всегда стоит сравнивать ключи вручную. Если вы создаете в локальной сети новый сервер для собственных нужд, сравнивать ключи, вероятно, не надо. (Тем не менее следует скопировать отпечаток, поскольку рано или поздно вам потребуется подключаться с удаленной машины. При этом придется проверить ключ.) Если к серверу будут подключаться много пользова-

телей, полезно разместить отпечатки на отдельной веб-странице. В любом случае надо решить, какой уровень секретности необходим.

Конфигурирование SSH

Все файлы для конфигурирования SSH в масштабе всей системы хранятся в каталоге */etc/ssh*. Рассмотрим их по одному.

Примечание

Предопределенные параметры SSH медленно изменяются по мере ужесточения требований к безопасности Интернета, поэтому здесь не приводятся значения по умолчанию для каждого параметра. Обратитесь к соответствующим файлам в вашей системе и уточните порядок настройки.

/etc/ssh/ssh_config

Файл */etc/ssh/ssh_config* управляет работой клиента ssh по умолчанию. Пользователи могут подменить значения в этом файле с помощью собственного *~/.ssh/config* или ключей командной строки, но данный файл дает администратору возможность задать разумные значения по умолчанию.

Примечание

Все значения, которые вы установили как клиент, могут быть подменены сервером. Например, вы можете запросить переадресацию X11, однако если сервер не предоставляет такую функцию, ее не удастся использовать.

`ForwardX11 yes`

Приложения X могут отображать данные на любой машине, а не только на той, на которой они запущены. Если необходимо запускать приложения X на удаленной машине и видеть данные на экране своей рабочей станции, установите значение `yes`.

`RhostsAuthentication yes`

Если учетная запись пользователя на удаленном сервере имеет соответствующий файл *.rhosts* на локальной машине, доверять ему. Почти всегда такая возможность отключается сервером в силу веских оснований; по существу, я не вижу причин, по которым ее надо было бы использовать.

`FallBackToRsh no`

Если подключение ssh завершается неудачно, клиент ssh попытается взамен установить соединение rsh, которое небезопасно. Если вы не может установить защищенное соединение, не подключайтесь вовсе – наверняка есть веские причины, по которым соединиться не удается! Если присвоить FallBackToRsh значение `no`, соединение rsh устанавливаться не будет.

`CheckHostIP yes`

Если этот параметр установлен, ssh-клиент будет автоматически сравнивать IP-адрес сервера с IP-адресом, указанным в файле *.ssh/known_hosts*.

Это помогает выявить злоумышленную имитацию IP-соединения и подмененные IP-адреса. Для отключения такой проверки установите значение `no`.

`StrictHostKeyChecking no`

Этот параметр – для страдающих паранойей. Если ему присвоить значение `yes`,[1] ssh-клиент откажется подключаться к хосту, чей ключ не представлен в *~/.ssh/known_hosts*. Также он откажется добавлять в файл *known_hosts* новые хосты, и придется добавлять их вручную.

`Port 22`

Это предопределенный порт для подключения к удаленному хосту. Значение этого параметра можно изменить для достижения большей безопасности за счет утаивания, но обычно так поступать нет смысла.

`Protocol 2,1`

Задает порядок, в котором версии протокола SSH предлагаются клиенту. Протокол отключен, если его нет в списке.

`Cipher blowfish`

SSH может использовать либо протокол 3DES, либо протокол Blowfish. Хотя Blowfish быстрее, это более новый протокол, а в мире криптографии «новее» не означает автоматически «лучше»! Тем не менее Blowfish до сих пор не поддается криптоанализу и имеет многообещающие перспективы. По умолчанию задан 3DES.

/etc/ssh/ssh_host_key и /etc/ssh/ssh_host_dsa_key

Эти файлы содержат секретные криптографические ключи системы. Файл DSA – для SSH2, а другой файл – для SSH1.

/etc/ssh/ssh_host_key.pub и /etc/ssh/ssh_host_dsa_key.pub

Это криптографические ключи системы, открытые для всех желающих. Криптографические системы с открытым ключом скомбинируют открытый и секретный ключи и создадут уникальный цифровой отпечаток. Файл DSA – для SSH2, другой файл – для SSH1.

/etc/ssh/sshd_config

Файл */etc/ssh/sshd_config* описывает сервисы, которые демон SSH предоставляет другим хостам. Клиент может затребовать любой протокол или сервис, но последнее слово остается за сервером. С помощью этого файла системный администратор может разрешить допустимые действия и запретить неприемлемые.

В следующих абзацах описаны параметры, которые содержатся в этом файле.

---

[1] В оригинале опечатка – «no». Еще о настройке ssh рассказано, например, по адресу *http://security.opennet.ru/man.shtml?topic=ssh&russian=0&category=&submit=Submit*. – *Примеч. науч. ред.*

`AllowGroups`

По умолчанию любой пользователь с приемлемой командной оболочкой может войти в систему, но если этот параметр установлен, то войти в систему смогут только пользователи указанных групп. Группы создаются в */etc/group* (см. главу 9). Указывая группы, разместите их в одной строке, разделив пробелами. (Допускается применение звездочки (*) в качестве группового символа, но нельзя использовать цифровые GIDs.) Перечисленные группы должны быть главными группами пользователей – группами, представленными в */etc/passwd*.

`AllowTcpForwarding`

Пользователи с доступом к SSH могут шифровать любой трафик между двумя хостами. Значение `no` запретит такую возможность. Однако если у пользователя есть доступ к командной оболочке, он сможет установить собственный механизм перенаправления данных (TCP port forwarder) и обойти это ограничение.

`AllowUsers`

Параметр позволяет явно перечислить пользователей, которым разрешена работа с данным сервером SSH. По умолчанию войти в систему может любой пользователь.

`Ciphers`

Тот, кто знает толк в криптографии, может выбрать порядок, в котором будут перебираться криптографические алгоритмы. Их надо перечислить в одной строке, разделив запятыми. Тем же, кто в криптографии не разбирается, лучше оставить значения по умолчанию.

`DenyGroups`

Данный параметр противоположен параметру `AllowGroups`, рассмотренному выше. Пользователи из этой системной группы не смогут войти в систему. Указанная группа должна быть их главной группой, то есть должна быть представлена в */etc/passwd*, а не просто в */etc/group*.

`IgnoreRhosts yes`

Файл *.rhosts files* – наследие тех дней, когда rlogin и rsh были общепринятыми стандартами UNIX. Они могут быть полезны при переходе с rsh на ssh, но уже устарели и опасны. Для того чтобы разрешить применение файлов *.rhosts*, установите значение `no`.

`KeepAlive`

Время от времени OpenSSH проверяет статус бездействующих соединений. Если другая сторона соединения не отзывается, сессия будет прервана и очищена. Такая проверка называется «keepalive». Ее применение может сопровождаться кратковременными неполадками в сети, с возможным прерыванием сессии SSH. Чтобы сессия SSH была по возможности открыта, установите значение `no`. Но без проверок «keepalive» осиротелые сессии SSH могут «болтаться» неделями – ваша сторона соединения может никогда не осознать, что компьютер на другой стороне был пе-

резагружен или даже воспламенился. Как правило, этот параметр лучше применять.

`PasswordAuthentication`

Определяет, каким образом пользователям разрешено применять пароли при входе в систему. Наиболее безопасна криптографическая аутентификация RSA или DSA, однако большинство пользователей не готовы к этому. (Пока.) Установите значение `yes`. Здесь аутентификация RSA и DSA обсуждаться не будет.

`PermitEmptyPasswords no`

Вредное назначение параметра почти в полной мере соответствует его названию. Не устанавливайте значение `yes`. Можете мне поверить, это ни к чему.

`PermitRootLogin no`

Определяет, сможет ли кто-либо напрямую через SSH войти в систему как root. Намного разумнее разрешить пользователям входить под своими именами и применять su(8), чтобы стать root. Тогда, если система будет взломана, у вас останется шанс выяснить, чья учетная запись при этом использовалась, и по крайней мере возложить вину на кого-либо. Это делу не поможет, но не исключено, что самочувствие ваше улучшится.

`UseLogin`

Если установить значение `yes`, то sshd будет совместим с login(1). Это позволит применять *login.conf* и все другие замечательные приемы входа в систему, описанные в главе 9.

`X11Forwarding`

Определяет, смогут ли клиенты передавать графику из программ X своим клиентским рабочим станциям. История безопасности X уходит корнями в далекое прошлое, поэтому многие администраторы отключают этот параметр без дополнительных размышлений. Однако существуют механизмы переадресации данных X11, разработанные сторонними производителями. Их может установить любой пользователь, имеющий доступ к командной оболочке. Кроме того, запрет переадресации данных X11 не запретит ручную переадресацию данных X по незашифрованному TCP/IP. По умолчанию параметр имеет значение `no`, но при наличии пользователей с доступом к командной оболочке его можно изменить на `yes`.

Системное время

Пользователи предполагают, что компьютер должен знать текущее время. Если база данных начнет вводить даты с запаздыванием на три часа или почта будет датироваться завтрашним днем, вы узнаете об этом довольно быстро. Для управления системным временем есть три инструмента: часовой пояс, tzsetup(8); средства сетевого протокола синхронизации времени (network time protocol), ntpdate(8); и ntpd(8). Прежде всего задайте свой часовой пояс.

Задание часового пояса

Часовой пояс нетрудно установить с помощью tzsetup(8) – программы, управляемой с помощью меню. Она выполнит соответствующие изменения в системе. В крупных компаниях можно задавать по умолчанию среднее время по Гринвичу, а в небольших организациях – местное время. Укажите географическое расположение вашей компании и выберите соответствующий часовой пояс.

Сетевой протокол синхронизации времени

При использовании сетевого протокола синхронизации времени (Network Time Protocol, NTP) каждая система устанавливает свое время по запросу. Клиенты могут принять это время и задать у себя то же значение, а могут определять время по значениям на других системах, вычисляя среднее значение. Среднее время – это лучший выбор на долгосрочную перспективу.

Для реализации NTP необходимы серверы времени. Многие серверы Интернета предоставляют сервисы времени, которые доступны всем пользователям. Серверы можно приблизительно разбить на два типа – уровня 1 и уровня 2 (Tier 1 и Tier 2).

Уровни серверов времени

Часы уровня 1 напрямую подсоединяются к высокоточным устройствам, ведущим хронометраж, таким как атомные часы. Им положено быть точными до абсурда. Если вам нужна такая точность, то приобретите атомные часы. Цены на них последнее время сильно упали, и сумма, которую надо будет уплатить за приличные атомные часы, выражается всего лишь в тысячах долларов. Кроме того, можно использовать другие системы, например радиочасы, – если допустимо отклонение значений, вызванное запаздыванием света. Если вам не нужна такая точность, обратитесь к серверам уровня 2.

Серверы NTP уровня 2 подпитываются серверами уровня 1, предоставляя сервисы времени во всеобщее пользование. Точность времени, сообщаемого этими сервисами, составляет доли секунды. Этого более чем достаточно для большинства приложений. Дополнительные поиски могут привести даже к серверам времени уровня 3, подпитываемым серверами времени уровня 2. Вам следует выбирать сервер уровня с наименьшим из возможных номеров, но любому серверу Интернета совершенно достаточно времени, сообщаемому сервером уровня 2 или 3.

Если поискать «NTP servers» в любом поисковике,[1] можно быстро найти свежайший список общедоступных NTP-серверов. Для каждого из ваших серверов выберите два ближайших NTP-сервера и запишите их имена и IP-адреса. Они понадобятся при настройке ntpdate и ntpd.

---

[1] Для таких целей удобнее всего использовать *Google.com* или *Yahoo.com*. – *Примеч. науч. ред.*

ntpdate

ntpdate(8) обращается к отдельному NTP-серверу, берет правильную дату и время, устанавливает системные часы и завершает свою работу. Как правило, программу ntpdate нужно выполнить только один раз, при начальной загрузке системы. Однако ее можно легко запустить из командной строки, задав имя или IP-адрес NTP-сервера:

```
# ntpdate kerberos.digex.net
30 Sep 17:30:44 ntpdate[616]: step time server 204.91.99.129 offset -35.707691 sec
#
```

Как можно видеть, системные часы шли вперед примерно на 35 секунд, однако теперь они синхронизированы с NTP-сервером *kerberos.digex.net*.

ntpdate при начальной загрузке

Программу ntpdate можно запустить при начальной загрузке. Задайте следующие параметры,[1] указав имя сервера времени:

```
ntpdate_enable="YES"
ntpdate_flags="timeserver.AbsoluteBSD.com"
```

Примечание

Не делайте этого на загруженном сервере с программами, восприимчивыми к времени, например серверами баз данных! Если все значения времени неожиданно увеличатся на час или два, администратор базы данных или клиенты будут вне себя.

Слабые места ntpdate

ntpdate проверяет время только один раз. Если в системе есть неполадки с оборудованием, системное время может постепенно выпасть из синхронизации. Это не так страшно для настольных операционных систем, но недопустимо для машин, которые должны работать без перезагрузки месяцы и годы. На «долгоживущих» системах необходимо либо запускать ntpdate регулярно, либо применять ntpd.

ntpd

ntpd(8) периодически сверяет системное время с серверами времени, входящими в определенный список. Опросив их, ntpd вычисляет разумное среднее значение и постепенно подгоняет системное время до соответствия среднему. Если значения на каком-то сервере времени существенно отличаются от остальных, то они не учитываются. Таким образом, вы получаете самое точное системное время, которое только возможно. При этом от того или иного сервера не требуется слишком много, а аппаратные средства находятся под контролем.

[1] В */etc/rc.conf*. – *Примеч. науч. ред.*

Версии ntpd

Первоначально демон протокола времени назывался ntpd, а улучшенная версия, входящая в состав FreeBSD, формально известна как xntpd. Поскольку первоначальная версия давно предана забвению, во всех нынешних системах xntpd называется ntpd. Не путайте эти версии.

Конфигурирование ntpd

Демону ntpd нужен конфигурационный файл */etc/ntpd.conf*. Вот пример:

```
driftfile /etc/ntp/drift
server 203.94.99.229
server 192.37.16.177
```

Начнем с самого легкого. Программе ntpd нужен временный файл (в предыдущем примере это */etc/ntp/drift*), причем она создаст этот файл самостоятельно, но не создаст необходимый каталог, то есть каталог */etc/ntp* придется создать самостоятельно. Далее представлены IP-адреса двух выбранных серверов уровня 2, к которым будет происходить обращение. Вот и все!

Серверы можно настроить на ретрансляцию обновлений времени по локальной сети Ethernet, разделяя информацию о времени со всеми другими серверами, в которых запущен ntpd. Звучит замечательно, однако такой подход не годится для серверов, которые общедоступны в Интернете.

Запуск ntpd

Создав */etc/ntp.conf*, наберите `ntpd`. Чтобы запускать ntpd при начальной загрузке, добавьте в */etc/rc.conf* следующую запись:

```
xntpd_enable="YES""
```

Демон inetd

Демон inetd обслуживает подключения к тем демонам, услуги которых запрашиваются сравнительно редко. Например, поскольку в большинстве систем нет устойчивого потока входящих запросов FTP, то и дополнительные издержки, вызванные тем, что демон FTP прослушивает порт, не имеют смысла, – ведь он бездействует 99,9% времени. Вместо этого за прослушивание порта FTP отвечает inetd. Когда поступает запрос FTP, inetd запускает демон FTP и передает ему запрос.

Кроме того, inetd обслуживает функции, которые настолько малы и так редко применяются, что их легче реализовать в inetd, нежели выделять для них отдельные программы. В состав таких функций входят discard (выполняет дамп данных, поступивших в «черную дыру» */dev/null*), chargen (генерирует поток символов) и т. д. По умолчанию они отключены, но при необходимости их можно активизировать. В типичную конфигурацию inetd включены сведения о многих стандартных сервисах UNIX, в том числе о telnet, ftp и pop3, а также о довольно многих малоизвестных протоколах.

/etc/inetd.conf

Рассмотрим файл */etc/inetd.conf*. Большинство демонов имеют отдельную конфигурацию для IP и IPv6. Таким образом, если вы не работаете с IPv6, все записи IPv6 можно игнорировать. Обратимся к одной из строк этого файла, задающей конфигурацию ftp:

```
❶ ftp   ❷ stream   ❸ tcp   ❹ nowait   ❺ root   ❻ /usr/libexec/ftpd   ❼ ftpd -l
```

Первое поле (❶) – это имя сервиса. Оно должно соответствовать имени в */etc/services*. По имени сервиса демон inetd определяет, какой порт TCP или UDP открыть.

Второе поле (❷) – тип сокета. Все соединения TCP имеют тип `stream`, а соединения UDP – тип `dgram`. Есть и другие возможные значения, однако их применение либо означает такое требование в документации той или иной программы, либо наверняка является ошибкой.

Третье поле (❸) – протокол. Это может быть `tcp`, `udp`, `tcp6` или `udp6` (протоколы `udp6` и `tcp6` предназначены для соединений IPv6). По мере распространения IPv6 и его интеграции с серверными программами в этом поле можно будет увидеть метки протоколов `udp46` и `tcp46`. Это означает, что данный демон может обслуживать соединение любого из двух видов.

Следующее поле (❹) указывает, должен ли inetd ждать, пока определенный сервер примет соединение, или должен запустить соответствующую программу и откланяться. Как правило, программы TCP используют `nowait`, а программам UDP необходимо значение `wait`. Если сервис применяет `nowait`, то можно задать максимальное количество соединений в секунду, которые разрешено обслуживать серверу. Для этого сразу после `nowait` надо добавить косую черту и соответствующее число, например `nowait/5`. Если это не сделано и на порт обрушился поток запросов, inetd запустит столько копий программы, сколько потребуется для обслуживания этих запросов. (Это простой способ поставить интернет-сервер на колени.)

Следующее поле (❺) сообщает, от чьего имени запускается демон. Некоторые демоны запускаются от имени специальных, выделенных пользователей. В следующих двух главах будут представлены конкретные примеры.

Шестое поле (❻) – полный путь к программе, которую inetd запускает, получив запрос на соединение. Если это сервис, включенный в состав inetd, будет указано `internal`.

Последнее поле (❼) задает команду, запускающую внешнюю программу, и необходимые ключи командной строки.

Конфигурирование программ в inetd

Файлу */etc/inetd.conf*, похоже, необходимо много информации, однако для добавления той или иной программы достаточно скопировать существующую строку и незначительно ее изменить. Например, рассмотрим реализацию тривиального сетевого сервиса Quote of the Day (QotD, цитата дня). Когда вы подключаетесь к порту QotD, сервер QotD возвращает случайную

цитату и разрывает соединение. В состав коллекции игр FreeBSD входит генератор случайных цитат fortune(1). Такой генератор – это все, что нужно для реализации сетевой программы на базе inetd. Далее программа fortune будет применяться для генерации случайных цитат.

Номер порта

Если в */etc/services* поискать «qotd», будет найдена ссылка на порт 17, то есть QotD работает на порту 17.

Сетевой протокол

Поскольку сервис QotD подразумевает обращение к сетевому порту и получение информаци в ответ, это TCP-сервис. (Альтернативный вариант, UDP, не подойдет, так как соединения UDP не подразумевают возврат чего-либо.) Следовательно, в конфигурации inetd необходимо указать протокол TCP. Для любого TCP-сервиса необходимо указать «nowait» в четвертом поле соответствующей записи *inetd.conf*.

Пользователь

Запустим команду, генерирующую цитаты, от имени root. В идеальном случае мы бы создали отдельную учетную запись для этого сервиса, но я не буду уделять много внимания этому примеру.

Путь

fortune находится в */usr/games/fortune*.

Запуск команды

Для запуска fortune ключи командной строки не нужны. (При желании можно задать `-o`, однако это не лучшая идея для общедоступного сервера.)

Простой пример конфигурации inetd.conf

Собрав все сведения вместе, добавим строку в */etc/inetd.conf*:

```
qotd    stream  tcp     nowait  root    /usr/games/fortune      fortune
```

Этот пример тривиален, однако другие строки и изменения в */etc/inetd.conf* не более сложны.

Защита inetd

Начинающие системные администраторы склонны полагать, что inetd – это единый сервис с монолитным состоянием защищенности. Нет ничего более далекого от истины. Сам по себе inetd защищен должным образом, однако его несправедливо обвиняют в проблемах программ, которым он перенаправляет запросы. Одни программы предоставляют небезопасные протоколы (например, telnet и ftp), а другие имеют подмоченную репутацию.

Тем не менее многие администраторы решительно отключают inetd. Другие отключают все сервисы, за исключением самого inetd, поскольку некоторые

программы предоставляют сервисы через inetd, а его включение позволяет проще устанавливать эти программы. Я рекомендую отключить inetd, если вы не предоставляете сервисы через inetd, а затем включить только их.[1]

Запуск inetd

Для запуска inetd из командной строки достаточно выполнить команду `inetd` как пользователь root. Либо можно предусмотреть автоматический запуск inetd при начальной загрузке, внеся изменения в */etc/rc.conf*:

```
inetd_enable="YES"
```

Изменение работы inetd

В переменной `inetd_flags` файла */etc/rc.conf* можно указать ключи, изменяющие поведение inetd при начальной загрузке. Ключи inetd, задаваемые по умолчанию, активизируют TCP Wrappers в соответствии с настройками в */etc/hosts.allow* (см. главу 8). Некоторые ключи представлены в следующей таблице:

Ключ	Описание
`-l`	Протоколирует все успешные соединения
`-c`	Устанавливает максимальное количество подключений в секунду, которые может обслуживать любой сервис. По умолчанию количество не ограничено (unlimited). Заметим, что unlimited не означает «infinite» (бесконечный) – количество подключений определяется возможностями аппаратных средств
`-C`	Устанавливает максимальное количество подключений к сервису, выполняемых с одного удаленного IP-адреса. По умолчанию количество не ограничено
`-R`	Устанавливает максимальное количество запусков сервиса, которые могут быть инициированы за одну минуту. Значение по умолчанию – 256. Если задать 0, будет разрешено неограниченное количество подключений
`-a`	Задает IP-адрес, к которому привязан inetd, обычно прослушивающий все доступные IP-адреса
`-w`	Применяет TCP Wrappers для внешних служб в соответствии с настройками в *hosts.allow* (см. главу 8)
`-W`	Применяет TCP Wrappers для внутренних служб в соответствии с настройками в *hosts.allow* (см. главу 8)

Рассмотрим пример. Если вы хотите применять TCP Wrappers, разрешить только два соединения в секунду с одного хоста и неограниченное количество соединений в минуту, укажите следующие ключи:

```
inetd_flags="-Ww -C 2 -R 0
```

[1] В любом случае не следует разрешать запуск inetd, если система не должна предоставлять сервисы, которые работают через него, и надо разрешать в */etc/inetd.conf* только необходимые сервисы в противном случае. – *Примеч. науч. ред.*

14

Почтовые службы

Одна из основных задач сервера Интернета состоит в том, чтобы передавать и получать электронную почту. FreeBSD – довольно мощный почтовый сервер, способный обслуживать миллионы сообщений в день. В этой главе обсуждается обработка потока электронной почты при взаимодействии «сервер-сервер», «клиент-сервер» и «сервер-клиент». Когда сервер взаимодействует с сервером или клиент посылает почту серверу, применяется стандартный протокол электронной почты SMTP. Когда клиент забирает свою почту с сервера, он применяет протокол POP3.

Обзор электронной почты

Большинство сообщений, передаваемых по электронной почте, создаются пользователями на настольных компьютерах. Чаще всего это PC под управлением Windows или Mac с Outlook, Eudora, Netscape или одной из подобных программ, однако почту можно посылать почти в любой операционной системе. Среди клиентов FreeBSD я отдаю предпочтение программе Mutt (*/usr/ports/mail/mutt*).

Клиент посылает сообщения серверу электронной почты. Почти каждая компания или провайдер услуг Интернета имеют по крайней мере одну выделенную почтовую систему. Почтовый сервер выполняет основную «санитарную» проверку сообщений, посылаемых клиентом, а затем пытается найти сервер, ответственный за это сообщение (см. «Нахождение верного почтового хоста» далее в этой главе). Почтовый сервер передает сообщение другому почтовому серверу.

Когда получатель проверяет наличие почты, клиентская программа соединяется с почтовым сервером, запрашивает все сообщения и загружает их на настольный компьютер. Если получатель отвечает, весь процесс повторяется в обратную сторону.

Где FreeBSD в самый раз

Серверная часть FreeBSD превосходна. Система FreeBSD, сконфигурированная должным образом, может обрабатывать тысячи сообщений в час. Если

вы приобрели хорошие аппаратные средства, система FreeBSD сможет получать и передавать свыше 40 000 сообщений в час. Это примерно 11 сообщений в секунду – в комплекте с бессвязным текстом, чудовищными графическими файлами и напыщенными страницами HTML, которые присоединяются к сообщениям.

Протокол электронной почты

Большая часть людей считает электронную почту волшебством, – вы нажимаете кнопку Send (Отправить), и сообщение «по эфиру» доставляется получателю. Однако на самом деле почту довольно легко послать вручную, не прибегая к клиентской программе. Этот прием можно использовать для устранения сложных неполадок или для того, чтобы похвастаться перед знакомыми. (Конечно, если кого-то из них можно поразить скучными трюками.)

Тестирование способности к взаимодействию

Определить, способен ли хост получать почту, можно с помощью telnet. Для этого надо указать, что вы хотите подключиться к серверному порту SMTP (25):

```
# telnet hostname 25
```

В первую очередь такой метод можно применять для того, чтобы определить, запущен ли почтовый сервер на той или иной системе. Подсоединимся к локальной машине[1] и проверим почтовую систему:

```
# telnet localhost 25
Trying ::1...
Trying 127.0.0.1...
telnet: connect to address 127.0.0.1: Connection refused
telnet: Unable to connect to remote host
#
```

Ясно. На моем лэптопе не запущен почтовый сервер. Попробуем другой сервер, с которого действительно можно получить ответ:

```
# telnet AbsoluteBSD.com 25
Trying 209.69.178.18...
Connected to AbsoluteBSD.com.
Escape character is '^]'.
220 AbsoluteBSD.com ESMTP Sendmail 8.9.3/8.9.3; Sun, 10 Jun 2001 17:23:15 -0400
(EDT)
```

Пожалуйста! Мы напрямую общаемся с почтовым сервером. Теперь мы знаем, что почтовая программа запущена. Данный сервер даже говорит нам, что в качестве агента передачи почты он применяет sendmail, а также сообщает местную дату и время.

---

[1] Вспомните сведения главы 5, согласно которым 127.0.0.1 всегда представляет локальный хост.

Наиболее загадочная часть вывода – первая часть отклика. В данном случае это число 220. Почтовый протокол предписывает, что каждый отклик сервера должен включать в себя как цифровой код, так и удобочитаемый ответ. Программа отправки обращает внимание только на первое число, а последующее описание предназначено для удобства восприятия отклика слабым, маленьким человеческим мозгом.

Общение с почтовым сервером

Теперь начнем общение с программой. Переговоры открываются с помощью команды `helo` и указания имени хоста, с которого вы подключаетесь:

```
helo turtledawn.AbsoluteBSD.com
```

Ответ сервера будет примерно таким:

```
250 AbsoluteBSD.com Hello pedicular.AbsoluteBSD.com [192.168.1.200], pleased to meet you
```

В ответе есть код отклика (250) и имя хоста, с которым вы общаетесь (*AbsoluteBSD.com*). Слово «hello» означает, что сервер готов общаться с вами. Далее представлено имя хоста, с которого вы подключаетесь. В данном случае DNS на сервере свидетельствует, что за адресом 192.168.1.200 скрывается *pedicular.AbsoluteBSD.com*.

Затем вы говорите почтовому серверу, от кого будет исходить ваше сообщение:

```
mail from: mwlucas@AbsoluteBSD.com
```

Сервер должен ответить, что вам позволено отправлять почту:

```
250 mwlucas@AbsoluteBSD.com... Sender ok
```

Если сервер не принимает почту с вашего адреса или хоста, он скажет об этом. Если все в порядке, вы вводите имя получателя в команде `rcpt to`:

```
rcpt to: mwlucas@AbsoluteBSD.com
250 mwlucas@AbsoluteBSD.com... Recipient ok
```

Теперь почтовый сервер знает отправителя и получателя. (Это наиболее типичное место, в котором отклоняется передача сообщения. См. раздел «Управление передачей» далее в этой главе.) Теперь все готово к отправке вашего сообщения. Выполните команду `data`:

```
data
354 Enter mail, end with "." on a line by itself
```

Теперь можно набирать любое сообщение. Согласно выводу команды, после подготовки сообщения надо ввести одну точку в отдельной строке. В следующем примере отправляются слова «Тестовое сообщение»:

```
Тестовое сообщение
.
```

После того как точка набрана в отдельной строке, почтовый сервер выдаст подтверждение о принятии сообщения к отправке:

```
250 RAA03288 Message accepted for delivery
```

Наберите `quit` для выхода:

```
quit
221 AbsoluteBSD.com closing connection
Connection closed by foreign host.
#
```

Такой метод можно применить как во благо, так и во вред. Как администратор вы можете проверять почтовую конфигурацию без возни с клиентской программой, которая может скрыть результаты теста.[1] Однако такой метод позволяет легко подделать сообщения, просто придумав какой угодно свой собственный обратный адрес `mail from`.[2]

Кто применяет почтовый протокол?

Почтовый протокол применяется при передаче сообщений с одного почтового сервера на другой. Кроме того, он применяется, когда почтовый клиент на настольном компьютере посылает сообщение своему серверу.

Управление ретрансляцией

Вообще говоря, почтовый сервер принимает почту, предназначенную для его локальных доменов, или почту, отправляемую с них. Продолжим рассмотрение примера с почтовым сервером *AbsoluteBSD.com*. Если данный сервер получает сообщение для пользователя с адресом в домене *AbsoluteBSD.com*, он принимает это сообщение. Если сообщение от пользователя с адресом в *AbsoluteBSD.com* предназначено для другого домена и если имеют место другие процедуры контроля доступа, то сервер принимает это сообщение. Если кто-либо, совершенно не связанный с *AbsoluteBSD.com*, пытается использовать данный почтовый сервер как ретранслятор (relay) для перенаправления писем на сторону, сервер отклонит такой запрос.

Пользователи, рассылающие незатребованные коммерческие предложения (*спам*), постоянно ищут почтовые серверы, которые позволяют передавать через себя электронную почту. Если на вашем сервере такая ретрансляция возможна, он становится потенциальным источником «макулатурной» электронной почты (junk email).

Примечание

Обязательно контролируйте прохождение почты через свою систему. Разрешая неограниченный поток писем через свои серверы, вы попадете в различные черные списки. Если не управлять ретрансляцией почты, можно потерять связь с 30% или 40% хостов Интернета.

---

[1] В некоторых кругах подделка сообщения другу есть дело чести. Уверяю вас, это не значит, что вы обязаны так поступать. Квалифицированный системный администратор может распознать поддельное сообщение с первого взгляда, просто проверив все заголовки письма.

[2] Легко догадаться, что все не так просто: у почтового сервера есть средства проверить, не лжет ли отправитель письма, и если лжет, отказать ему в передаче письма. Далее рассказано, как настраивать эти средства. – *Примеч. науч. ред.*

Итак, что представляет собой «контроль доступа»? Одна из самых распространенных составляющих – ограничение IP-адресов, с которых через вашу систему можно посылать почту на любые адреса. Если отправку писем через данный сервер вы разрешите только пользователям локальной или корпоративной сети, посторонние лица уже не смогут использовать ваш сервер для передачи «макулатурной» почты.

Если вы предоставляете возможность коммутируемого соединения с сервером, вы можете сконфигурировать сервер так, чтобы разрешить ретрансляцию с этих IP-адресов. Возможно, кто-либо заключит с вами договор на коммутируемое подключение, чтобы отправлять через вас «макулатурную» почту. Лучше всего, если в договоре на предоставление услуг будут не только запрещены такие действия, но и предусмотрены штрафные санкции, компенсирующие поток негодования в ваш адрес.

Черные списки отправителей «макулатурной» почты

Вы можете спросить: «Как используется черный список серверов, с которых отправляют «макулатурную» почту?» Применение одного из сервисов, ведущего список таких серверов, предоставляет очень эффективный способ урезать объем получаемой «макулатурной» почты, однако такой сервис может заблокировать легитимный трафик. Следовательно, необходимо по крайней мере знать, применяете ли вы такой сервис. Обычно услуги таких серверов предоставляются по подписке и предусматривают заключение контракта.

Самая большая служба, блокирующая «макулатурную» почту, – Realtime Blackhole List, или RBL (*http://mail-abuse.org*). В большинстве почтовых серверов предусмотрена проверка сайтов на соответствие RBL. Обратитесь к веб-странице сайта, предоставляющего такую службу, и изучите, как интегрировать его функции в свой почтовый сервер.

Почтовые программы

В течение многих лет в состав UNIX входит почтовый сервер sendmail. Это огромная, малопонятная, бестолковая программа, которая совершенно устрашает начинающих администраторов. Многие опытные администраторы UNIX также находят ее огромной, малопонятной, бестолковой и совершенно устрашающей. Взгляните на */etc/mail/sendmail.cf* и вы получите представление о базовом конфигурационном файле sendmail.

Посмотрели? Хорошо, теперь можно перевести дух. FreeBSD поддерживает программы, альтернативные sendmail. Они просты в настройке, более эффективны и безопасны. Если вы не должны предоставлять почтовые услуги в старом стиле, в sendmail нет необходимости.

Кому нужен sendmail?

Древнее (для Интернета) изречение гласит: «sendmail сложен потому, что сложен мир». Это правда. Кроме того, sendmail хорошо протестирован, а член ко-

манды FreeBSD, отвечающий за поддержку sendmail, также является разработчиком этой программы. sendmail необходим, если почта передается по BITNET, UUCP и другим загадочным протоколам. Сегодня в Северной Америке их почти никто не применяет. Такие службы применяются в других частях света, где полоса пропускания недешева, а аппаратные средства еще дороже.

sendmail даже нарушает одно из важнейших правил UNIX, утверждающее, что множество небольших инструментов можно объединять по своему усмотрению. sendmail – огромная и монолитная программа. Альтернативные средства меньше; они собраны из нескольких индивидуальных программ.

Если ваши почтовые сервисы обслуживают только Интернет (подобно большинству известных мне почтовых серверов), вам не нужен sendmail. Если вы предоставляете услуги большому количеству клиентов, в какой-то момент без sendmail уже не удастся обойтись. Решение на основе sendmail можно предоставить для того или иного клиента, а остальным предоставлять услуги на более простой, легкоуправляемой почтовой платформе.

Лично мне нравится sendmail. И я очень горжусь своей способностью вручную редактировать */etc/mail/sendmail.cf*. А еще я знаю парня, который гордится способностью разбивать пивные кружки о свой лоб. Однако хвастаться в обоих случаях нечем.

Альтернатива sendmail

Две самых популярных альтернативы sendmail – почтовые серверы qmail и postfix. По сравнению с sendmail оба меньше и проще в обеспечении безопасности. Их проще конфигурировать. Лицензия postfix написана в стиле BSD, а qmail имеет ограничения на использование, модификацию и передачу третьим лицам. Все остальное в них одинаково, отличаются только лицензии; мы будем рассматривать postfix. Поскольку postfix может обслуживать до миллиона почтовых сообщений в день на обычных аппаратных средствах, эта программа, безусловно, удовлетворит ваши нужды.

Установка postfix

postfix можно установить аналогично любым другим программам – через «порт» или пакет. Однако в установке postfix есть несколько дополнительных шагов, которые различаются в зависимости от версии FreeBSD. Я рекомендую установить postfix из «порта» и следовать прилагаемым инструкциям.

Части postfix

В отличие от sendmail, postfix состоит из нескольких маленьких частей. Одна часть обслуживает получение почты, приходящей по сети. Другая отвечает за доставку почты в личные почтовые ящики, а третья передает почту, находящуюся в очереди. Для эффективной работы с postfix необходимо по крайней мере иметь понятие о почтовых компонентах. Не беспокойтесь, если вы не понимаете назначение всех этих функций, – они будут рассмотрены последовательно.

master

Демон master надзирает за всеми другими частями postfix. Согласно его предписаниям выполняются другие программы. Если какая-нибудь программа не запускается, проверьте конфигурацию демона master.

smtpd

Программа smtpd получает почту из сети. Она проверяет, действительно ли то или иное письмо является обычным сообщением или же это атака.

sendmail

Программа sendmail обслуживает почту, созданную локально. Многие программы UNIX предполагают возможность сделать дамп своего вывода в файл, называемый «sendmail», и затем отправить его по почте. Для минимизации возможного ущерба в состав postfix включена программа, которая работает как sendmail, но доставляет почту системе postfix (в каталог *maildrop*, который мы обсудим позже).

pickup

Программа pickup берет сообщения из каталога *maildrop*, проверяет на наличие ошибок и выталкивает их из postfix.

cleanup

Программа cleanup получает сообщения из всех остальных источников. При необходимости она добавляет заголовок From:. Кроме того, она должным образом преобразует почтовые заголовки и помещает сообщения во входящую очередь.

queue manager (qmgr)

Входящие и исходящие письма находятся в очередях. Администратор очереди (qmgr) проверяет каждое сообщение, определяет пункт назначения и передает его соответствующему агенту доставки.

trivial-rewrite

Демон trivial-rewrite анализирует адреса (являются они локальными или же удаленными) и перезаписывает заголовки надлежащим образом.

local

Локальный агент доставки помещает сообщения в локальные почтовые ящики пользователей. Если вы заменяете sendmail и предписываете пользователям применять файлы *.forward* или программу procmail, используйте агент доставки.

virtual

Виртуальный агент доставки обеспечивает доставку писем в локальные почтовые ящики пользователей, но не поддерживает файлы *.forward* или программу procmail. Однако он обслуживает виртуальные домены. Такой агент – лучший выбор, если вы запускаете типичный сервер Интернета, с которого клиенты загружают почту на свои персональные компьютеры.

smtp client

Программа smtp client принимает почту от администратора очереди и пытается доставить ее на удаленные хосты (другие домены).

Конфигурирование postfix

Конфигурационные файлы postfix хранятся в */usr/local/etc/postfix*. Здесь можно найти массу файлов-примеров, но в основном они представляют собой текстовые версии страниц руководства. В первую очередь надо обратить внимание на файлы *main.cf* и *master.cf*.

В файле *master.cf* содержатся указания для программы master о том, как обслуживать другие демоны, за которые она отвечает. Этот файл можно настраивать, но эта возможность наверняка не пригодится. Программа postfix работает быстро и достаточно эффективно с настройками по умолчанию.

main.cf управляет обработкой почты. Он сообщает программе postfix, куда посылать письма различных типов, какую почту принимать и как работать в целом. Этот файл необходимо конфигурировать.

Здесь не будут рассматриваться все параметры. К каким-то из них никогда не надо обращаться, если вы начинающий администратор почты. Другие параметры (например, путь к программам) очевидны. Как правило, параметры представляют собой переменные. Например, следующая строка в *main.cf* определяет переменную `$myhostname`:

```
myhostname = mail.AbsoluteBSD.com
```

В этом операторе настройки можно задать другие значения переменной `$myhostname`. Тогда переменная будет надлежащим образом изменена по всей системе. В «порту» заданы разумные значения по умолчанию, однако наверняка вам потребуется дополнительная тонкая настройка.

Итак, без лишних слов приступим к рассмотрению конфигурационных операторов, необходимых для базового обслуживания электронной почты.

```
myhostname = mail.AbsoluteBSD.com
```

Переменная `myhostname` содержит имя хоста по умолчанию, которое изначально берется непосредственно из операционной системы. Эта переменная применяется во многих операциях. Ее значение изменять не надо, но это может пригодиться для какого-нибудь трюка. Например, можно задать имя машины, являющейся частью сети, но предназначенной для отдельного клиента с другим именем домена. Кроме того, значение `myhostname` можно изменить, если вы находитесь за брандмауэром и хотите скрыть имя своего хоста.

```
mydomain = AbsoluteBSD.com
```

Переменная `mydomain` подобна `myhostname`; `mydomain` — это имя домена для данного хоста. Оно образуется отсечением первого слова в имени хоста. Имя домена может потребоваться установить вручную, если у вас необычное имя хоста.

```
myorigin = $mydomain
```

Переменная `myorigin` определяет, какой источник будет указываться в качестве отправителя в исходящих письмах. По умолчанию берется значение `$myhostname`. На главном почтовом сервере, возможно, понадобится устано-

вить значение $mydomain. В конце концов, лучше, чтобы в поле адреса стояло выражение «username@domain.com», а не «username@mail.domain.com».

На других машинах, которые отправляют почту, подойдет значение по умолчанию. Вероятно, все эти удаленные машины будут автоматически отправлять вам сводки, и поэтому указание неправильного адреса отправителя может лишь запутать вас.

```
mydestination = $myhostname, localhost.$mydomain
```

Переменная mydestination задает домены и имена хостов, с которых машина будет принимать почту. По умолчанию почта принимается для системы, указанной в $myhostname, и для localhost. Например, *mail.AbsoluteBSD.com* примет почту для *mail.AbsoluteBSD.com* и *localhost.AbsoluteBSD.com*.

Значение mydestination, задаваемое по умолчанию, подходит для отдельной машины, но для корпоративного почтового сервера надо добавить дополнительные хосты. Пример, поставляемый с postfix, служит хорошей отправной точкой:

```
mydestination = $myhostname, localhost.$mydomain, $mydomain,
            mail.$mydomain, www.$mydomain, ftp.$mydomain
```

В этот список может понадобиться добавить другие важные машины сети, например серверы имен.

Такая настройка обеспечит работу postfix.

Ограничение передачи почты

Самый простой способ управлять передачей почты в postfix – ограничение IP-адресов. Оператор mynetworks в *main.cf* определяет, какие клиенты могут передавать почту через сервер:

```
mynetworks = subnet
```

Значение, задаваемое по умолчанию, подходит для небольшого офиса, однако при предоставлении почтовых услуг в Интернете надо добавить дополнительные значения. Ключевое слово subnet разрешает postfix передавать почту от компьютеров, находящихся в одной сети с почтовым сервером. Запустите ifconfig -a и получите текущий адрес подсети. Для указания дополнительных сетей укажите в списке их IP-адреса. Различные подсети разделяются запятыми.

```
mynetworks =  192.168.141.128/28, 127.0.0.0/8
```

Если вы не можете передать почту с клиентской машины, убедитесь, что ее адрес указан в $mynetworks.

```
relaydomains = $mydestination
```

Кроме того, для управления передачей почты можно применять доменное имя. Для этого в *main.cf* есть опция relaydomains. В данном примере postfix передаст почту, если она исходит с хоста, указанного в списке $mydestination, или адресована такому хосту.

Если вы применяете виртуальные домены (см. раздел «Виртуальные домены» далее в этой главе), postfix также будет передавать почту этих доменов.

Ретрансляция почты на главной машине

Для того чтобы все машины передавали почту через главный почтовый сервер, можно применить ключевое слово `relayhost`. Серверов, передающих почту от случая к случаю, может быть и десяток, однако все взаимодействие с Интернетом должен обслуживать главный почтовый сервер. (Это очень распространенная конфигурация.) Присвойте переменной `relayhost` имя или IP-адрес вашего почтового сервера:

```
relayhost = mail.AbsoluteBSD.com
```

Почтовые псевдонимы

Файл */etc/mail/aliases* содержит инструкции переадресации почты, посылаемой указанным пользователям. Файл псевдонимов берет начало из sendmail, но его понимают и многие другие почтовые программы. Каждая строка начинается с имени псевдонима; за ним следует двоеточие и список пользователей, которым будет перенаправляться почта.

Перенаправление почты от одного пользователя другому

Многие администраторы предпочитают, чтобы почта, адресованная пользователю root, перенаправлялась на их почтовый ящик. Такая схема реализована в следующем примере:

```
root:     mwlucas@absolutebsd.com
```

Перенаправление почты несуществующему пользователю

Многие почтовые адреса не имеют соответствующих им персональных учетных записей. Например, стандарты Интернета требуют, чтобы в каждой системе, отправляющей почту, был почтовый адрес «postmaster». Но никто не заводит для этого отдельную учетную запись – почта с этих адресов перенаправляется на почтовые ящики действительных пользователей:

```
postmaster:     root
```

Файл *aliases* уже содержит различные стандартные псевдонимы для адресов, которые обычно должны присутствовать на сервере. Изучите этот файл и обновите его в соответствии с задачами своей системы.

Псевдонимы списков рассылки

Также можно перечислить нескольких пользователей, создавая небольшие локальные списки рассылки. Такой подход недостаточно масштабируем, если пользователей много, однако он годится для решения небольших задач.

```
sales:     mwlucas, bpollock, sales@nostarch.com
```

Перенаправление почты в файлы

В файле псевдонимов особый интерес представляет возможность перенаправлять почту по адресу, который не является почтовым ящиком пользователя. Если указать имя файла, сообщение будет добавлено в этот файл. Для постоянного протоколирования всей почты можно задать такую строку:

```
username:     /var/log/username-log, username
```

Перенаправление почты программам

Письма также можно посылать программам, которые выполняют их автоматическую обработку. Задайте имя программы, указав перед ним символ конвейеризации (|). Например, если вы написали сценарий, обрабатывающий входящую почту, для перенаправления писем подойдет такая строка:

```
orders:          |/usr/local/bin/process-orders.pl
```

Списки в файлах псевдонимов

Наконец, в файл псевдонимов можно включать другие файлы. Это дает пользователю возможность модифицировать псевдоним по своему усмотрению.

```
clientlist:     include:/usr/home/salesdude/clientaddresslist
```

В этом примере файл */usr/home/salesdude/clientaddresslist* – просто список почтовых адресов, по одному в каждой строке. Он позволит продавцу поддерживать список клиентов, не отвлекая вас при каждом добавлении нового адреса.

Активизация изменений псевдонимов

Единственное предостережение – в этой простой системе файл */etc/mail/aliases* не анализируется при получении каждого письма. На самом деле файл *aliases* применяется для создания небольшого файла базы данных (*/etc/mail/aliases.db*), который postfix применяет для маршрутизации почты. Доступ к двоичной базе данных занимает намного меньше времени, чем сканирование текстового конфигурационного файла. Это особенно важно в системах с ограниченной мощностью процессора, а также в системах, обслуживающих большой объем электронной почты.

Редактируя файл псевдонимов или любой файл, входящий в его состав, необходимо запустить newaliases и пересобрать эту базу данных. Программу newaliases можно безопасно запускать из cron; пользователи, поддерживающие включаемые файлы, не увидят свои изменения до тех пор, пока не выполнится задание, запущенное из cron. Однако большинство пользователей согласятся на этот вариант, если будут знать, чего ожидать.

Протоколирование почты

Почти все почтовые программы помещают сообщения о своей работе в */var/log/maillog*. Если вы хотите знать, чем занимается сервер, проверьте этот файл. Помните, можно запустить `tail -f /var/log/maillog` и наблюдать проис-

ходящие события по ходу их возникновения. Тип сообщений, которые попадают в файл протокола, зависит от применяемого почтового сервера.

Виртуальные домены

Одна из основных причин популярности FreeBSD заключается в наличии возможности поддерживать множество доменов на одном сервере. Большинству людей, желающих организовать интернет-сервис на базе доменного имени, надо совсем немного: веб-страница и почтовый адрес. Одна машина FreeBSD способна обслуживать сотни простых веб- и почтовых сайтов, обратившись к волшебству виртуальных доменов.

Концепция виртуального домена проста: это дополнительное имя сервера. Сервер конфигурируется для обслуживания веб-запросов и почты, относящихся к данному домену. Веб-сервис будет рассмотрен при обсуждении веб-сервера Apache в главе 15. Сейчас рассмотрим электронную почту.

Для получения возможности применять виртуальные домены добавьте в *main.cf* следующую строку:

```
virtual_maps = /usr/local/etc/postfix/virtual
```

Такая строка сообщает программе postfix, где искать информацию о виртуальных доменах. Информация о виртуальных доменах хранится в «карте» (map), в которой виртуальным пользователям поставлены в соответствие реальные пользователи системы. По умолчанию почта доставляется пользователю, имя которого в системе соответствует почтовому имени в письме. Например, пользователь mwlucas на *AbsoluteBSD.com* имеет электронный адрес *mwlucas@AbsoluteBSD.com*. Если бы я хотел предоставить тому или иному заказчику адрес *mwlucas@vanhornefabrication.com* и обслуживать почту этого домена на том же сервере, то письма заказчика по умолчанию попадали бы в мой почтовый ящик. Это плохо. Почтовая карта виртуальных доменов предписывает postfix помещать письма, приходящие на этот адрес, в другой почтовый ящик.

Карты виртуальных доменов

Формат файла виртуальных доменов очень прост:

```
domainname.com
postmaster@domainname.com     system-user1
user2@domainname.com          system-user2
user4@domainname.com          system-user3
```

Во-первых, вам требуется имя домена, на базе которого предоставляется сервис. Затем надо указать действительный почтовый адрес, а также учетные записи пользователей или почтовые адреса, на которые будет перенаправляться почта. Например, при организации виртуального домена для *AbsoluteBSD.com* можно создать такой файл:

```
AbsoluteBSD.com
postmaster@AbsoluteBSD.com     mwlucas
sales@AbsoluteBSD.com          sales@nostarch.com
```

```
questions@AbsoluteBSD.com      mwlucas
refunds@AbsoluteBSD.com        /dev/null
```

Сообщения для *postmaster@AbsoluteBSD.com* перенаправляются пользователю mwlucas. Письма на адрес *sales@AbsoluteBSD.com* перенаправляются на совершенно другой домен. Почта на *questions@AbsoluteBSD.com* также уходит пользователю mwlucas, а письма на *refunds@AbsoluteBSD.com* попадают в */dev/null*.

Подобно многим другим конфигурационным файлам UNIX, таблица виртуальных доменов на самом деле хранится в небольшом файле базы данных. После редактирования файла виртуальных доменов надо обновить базу данных с помощью postmap(8):

```
# postmap /usr/local/etc/postfix/virtual
```

Если заставить postfix перечитать конфигурационные файлы, изменения вступят в силу лишь спустя некоторое время. Таким образом, мы плавно переходим к следующей теме.

Команды postfix

В postfix есть несколько команд, упрощающих управление почтовым сервером. Рассмотрим основные команды.

`postconf`

Выводит список всех конфигурационных параметров postfix, включая значения всех переменных.

`postfix check`

Проверяет конфигурацию postfix и находит особенно серъезные проблемы.

`postfix start`

Запускает систему postfix.

`postfix stop`

Эта команда (не спешите...) останавливает postfix.

`postfix reload`

Заставляет postfix проверить все конфигурационные файлы на наличие изменений. Эта проверка так или иначе выполняется каждые несколько минут, однако принудительное перечитывание полезно, если надо торопиться.

Нахождение верного почтового хоста

Итак, мы знаем, как передавать и получать почту от сервера к серверу и от клиента к серверу. Как почтовый сервер узнает, какому удаленному серверу надо передать то или иное письмо?

Когда почтовый сервер получает письмо для доставки на удаленный домен, он обращается к DNS. В записи DNS для данного домена почтовые серверы

перечислены в виде записей «MX» (см. главу 12). Почтовый сервер пытается доставить письмо почтовому серверу с наименьшим номером приоритета. Если предпочтительный почтовый сервер не отвечает, происходит обращение к серверу с наименьшим номером приоритета из оставшихся. Серверы последовательно перебираются по возрастанию номера приоритета, пока один из них не сможет принять письмо либо список серверов не будет исчерпан.

Недоставленная почта

Если сообщение не удалось доставить, сервер помещает его в очередь. Время от времени он пытается снова доставить сообщение. Если сообщение не удается доставить в течение пяти дней, оно возвращается отправителю как «недоставленное» (undeliverable).

POP3

POP3 – это протокол, применяемый почтовыми клиентами настольных компьютеров для получения писем с сервера. Передача почты от клиента на сервер осуществляется через SMTP, подобно передаче почты между серверами.

Установка POP3

Самый распространенный демон POP3 – qpopper (*/usr/ports/mail/qpopper*). Данная программа берет свое начало в BSD, а в последнее время ее поддерживает Eudora. Демон qpopper можно установить из пакета или «порта».

Qpopper вызывается из inetd. При установке из «порта» или пакета появится сообщение, поясняющее, как отредактировать */etc/inetd.conf* для поддержки qpopper. Предлагаемый пример подходит для того, чтобы применить его по умолчанию; позднее будет рассмотрена тонкая настройка конфигурации.

Тестирование POP3

POP3 может работать в режиме с шифрованием и без шифрования. Тем, кто не может выполнять в уме криптографические вычисления, будет нелегко проверить POP3 с шифрованием. Однако протестировать POP3 без шифрования не составит труда. Проверка работы qpopper поможет выявить, где имеют место неполадки: на сервере или на клиенте.

Вначале с помощью telnet подключимся к порту 110 на сервере.

```
# telnet magpire.AbsoluteBSD.com 110
Trying 192.168.1.222...
Connected to magpire.AbsoluteBSD.com.
Escape character is '^]'.
+OK Qpopper (version 4.0.3) at magpire.AbsoluteBSD.com starting.
<3915.992459999@magpire.AbsoluteBSD.com>
```

Примерно такие строки должны появиться при подключении.

Аутентификация для POP3

Подключившись с помощью telnet, представьтесь серверу POP3. Для этого выполните команду `user`:

```
user mwlucas
+OK Password required for mwlucas.
```

Теперь с помощью команды `pass` введите свой пароль. Пароль будет показан на экране открытым текстом. Убедитесь, что никто не стоит у вас за спиной в это время!

```
pass YourPasswordHere
+OK mwlucas has 1 visible message (0 hidden) in 500 octets.
```

Просмотр почты

У меня одно сообщение! Это странно; я не получаю почту в этой системе. Для просмотра сообщения применим команду `retr`, указав номер сообщения.

```
retr 1
+OK 500 octets
Return-Path: <mlucas@gltg.com>
Delivered-To: mwlucas@magpire.AbsoluteBSD.com
Received: from turtledawn (turtledawn [192.168.1.200])
        by magpire.AbsoluteBSD.com (postfix) with SMTP id D51998041C
        for <mwlucas@magpire.AbsoluteBSD.com>; Fri,  8 Jun 2001 14:48:59 -0400 (EDT)
Message-Id: <20010608184859.D51998041C@magpire.AbsoluteBSD.com>
Date: Fri,  8 Jun 2001 14:48:59 -0400 (EDT)
From: mlucas@AbsoluteBSD.com
To: undisclosed-recipients:;
X-UIDL: $ld¨!9>2¨!P?)¨!JlU¨!
test
```

Правильно. Я создал это сообщение, когда объяснял, как тестировать почтовый сервер.

С помощью таких тестов можно убедиться, что с POP3 все в порядке. Если получены иные результаты, необходимо продолжить исследование.

Протоколирование POP3

Если запустить `qpopper` с ключом `-s`, все операции будут протоколироваться с использованием syslog от имени источника local0 и уровнем серьезности notice (см. главу 19). По умолчанию события протоколируются в */var/log/messages*, однако файл можно задать по своему усмотрению.

Режимы POP3

POP3 можно реализовать тремя способами: режим по умолчанию, APOP и SSL (pop3ssl).

POP3 по умолчанию

Пример POP3 по умолчанию рассмотрен ранее. Это действующий вариант, но он небезопасен. С помощью анализатора пакетов недоброжелатель может перехватить имя пользователя и пароль, словно он стоит у вас за спиной. Это типичный протокол, предлагаемый провайдерами услуг Интернета.

APOP

APOP предлагает механизм надежной аутентификации, но подразумевает дополнительные операции. Как клиент, так и сервер вычисляют «общий секрет» (shared secret), основанный на пароле и другой информации, например на текущем времени. Это значение клиент посылает на сервер. Если секретное значение совпадет с вычислениями на сервере, доступ будет предоставлен.

APOP может быть хорошим выбором для вашего сервера: он выпущен немного раньше, чем pop3ssl, и его поддерживают многие клиентские программы. Хотя информация об аутентификации защищена, сама почта не находится в безопасности.

Pop3ssl

Pop3ssl – это новейшая версия протокола POP3, обеспечивающая туннель для аутентификации и передачи данных по соединению SSL. На сегодня это самый безопасный вид сервиса POP3.

Рассмотрим каждый вид POP3 по очереди. Для применения APOP или pop3ssl так или иначе необходимо провести базовую настройку POP3.

Вопросы перед конфигурированием qpopper

Перед конфигурированием qpopper необходимо ответить на два вопроса: какие пользователи будут в системе и будут ли применяться локальные почтовые клиенты?

Типы пользователей

Если вы предоставляете корпоративный почтовый сервис с помощью qpopper, то в конечном итоге отвечаете за настройку клиентских программ (или, в лучшем случае, работаете с пользователями, которые должны их настроить). Вы можете настоять на выполнении таких условий, как «все пользователи должны набирать свои имена строчными буквами» или «почта должна оставаться на сервере». Кроме того, можно настоять на применении APOP или pop3ssl вместо режима POP3 по умолчанию.

Если сервисы предоставляются сотням или тысячам пользователей, то конфигурация должна допускать больший спектр ошибок пользователей и работать с более широким множеством клиентских почтовых программ. Нельзя удержать пользователей, настаивая на применении определенных почтовых программ вместо тех, к которым они уже привыкли!

Локальные почтовые клиенты

Некоторые пользователи читают почту локально на самом сервере. Для этого они применяют почтовые клиенты на базе UNIX, такие как mutt(1) или pine(1). Данные программы изменяют файлы почтовых ящиков пользователей непосредственно на самом сервере.

Если qpopper может с уверенностью предположить, что содержимое ящика не изменяется кем-то еще в данный момент, он способен провести оптимизацию, которая значительно увеличит производительность. Это не так важно для системных администраторов – большинство из них не применяют POP3, а выбирают ssh или локальные почтовые клиенты. Однако продвинутые пользователи могут задействовать обе возможности. Если вы не комбинируете локальные почтовые программы и POP3, то можете оптимизировать qpopper.

Конфигурация qpopper по умолчанию

Как было показано ранее, «сырая» установка qpopper обеспечивает основную функциональность POP3. Пользователи смогут подключаться к серверу и загружать свою почту. Однако для улучшения производительности можно предпринять различные шаги, при этом дополнительная настройка не представляет сложности.

Более ранние версии qpopper конфигурировались исключительно с помощью ключей, задаваемых в */etc/inetd.conf* (в поле командной строки). Все шло хорошо, пока qpopper был простой программой, которая поддерживала только POP3 по умолчанию. Однако по мере распространения APOP и pop3ssl конфигурирование в командной строке становились все менее и менее удобным. Если аргументы командной строки занимают две или три строки экрана, то действительно пора перейти к использованию файла настройки.

Хотя простому POP3 конфигурационный файл не нужен, рассмотрим его применение.

Файлы настройки и inetd

В системе FreeBSD конфигурационный файл qpopper обычно располагается в */usr/local/etc/qpopper*. По умолчанию доступ к каталогу qpopper разрешен только пользователю root. В улучшенных реализациях qpopper в этом каталоге также хранятся базы данных пользователей и сертификаты безопасности.

Для того чтобы qpopper считывал значения параметров из данного файла, применяется ключ `-f`. Все остальные изменения можно выполнять в конфигурационном файле, не обращаясь к */etc/inetd.conf*. Тогда запись в *inetd.conf* будет такой (безотносительно к ширине страницы она представляет собой одну строку):

```
pop3 stream tcp nowait root /usr/local/libexec/qpopper qpopper -s -f /usr/
local/etc/qpopper/qpopper.conf
```

qpopper.conf

Итак, теперь система знает, как запускать qpopper, и можно создать конфигурационный файл. Каждый оператор настройки в *qpopper.conf* размещается в отдельной строке, а перед ним ставится слово «set». Любой из этих параметров можно комбинировать с APOP и pop3ssl.

Режим qpopper

Самый важный параметр – режим работы qpopper. Следующий параметр определяет, будет ли qpopper принимать пароли, передаваемые открытым текстом (как в примере тестирования qpopper, описанном ранее):

```
set clear-text-password = default
```

Здесь возможны несколько значений. Рассмотрим самые распространенные.

По умолчанию qpopper проверяет, может ли пользователь применять APOP. Если да, то передача паролей открытым текстом не разрешена. Если поддержки APOP нет, пароли могут передаваться открытым текстом. Это подходит для стандартных сервисов.

Значение `always` означает, что qpopper принимает пароли, передаваемые открытым текстом, даже если клиентская программа может работать с APOP. Такая настройка может понадобиться компании-провайдеру; хотя вы предпочитаете APOP, некоторые клиентские программы такую возможность просто не поддерживают.

Значение `never` означает, что пароли нельзя передавать открытым текстом даже в случае pop3ssl. Для получения почты следует применять APOP.

Значение `tls` означает, что передавать пароли открытым текстом допустимо при защищенном соединении (таком как SSL). В конце концов шифруются все передаваемые данные!

Реализация APOP и POP3 по SSL будет рассмотрена позднее (в разделах «APOP Setup» и «Configuring Pop3ssl» соответственно).

Регистр имени пользователя

Из множества пользователей системы кто-нибудь обязательно будет набирать свое имя прописными буквами. В конце концов, так набирают имена в кинофильмах! Имена пользователей, получаемые от клиентских программ, можно преобразовывать в нижний регистр до того, как qpopper попытается их аутентифицировать. Для этого служит следующая строка:

```
set downcase-user            = true
```

Этот параметр снизит количество обращений в службу технической поддержки.

Обслуживание почтового ящика

Клиент POP3 может либо копировать все сообщения с сервера, либо загружать к себе и удалять с сервера, либо удалять часть сообщений, а остальные оставлять. Два первых варианта (все оставлять и все удалять) типичны

с точки зрения функциональности почтового сервера. Третий вариант, смесь сохранения и удаления, требует больших усилий. Такой режим можно выбрать с помощью следующей строки:

```
set server-mode              = false
```

Установка режима сервера (server mode) предполагает, что клиент либо сохраняет всю почту на сервере, либо удаляет ее. За счет этого qpopper будет работать намного быстрее, а объем дискового ввода/вывода на сервере снизится. Если включить режим сервера, эффективность qpopper значительно повысится.

Однако активизация режима сервера налагает определенные обязательства. Демон qpopper предположит, что почту клиентам доставляет только он. Именно здесь возникают трудности с «перемешиванием локальных почтовых агентов и POP3». Если с помощью локальной программы вы проверяете почтовый ящик того или иного пользователя и читаете его письма, а в это время он скачивает данные письма по POP3, они могут повредиться. Это вам ни к чему. Если вы не комбинируете POP3 и локальные почтовые клиенты, а также не читаете письма пользователей, значение `true` будет безопасным выбором и позволит улучшить производительность.

Снижение дисковой активности

Если установить следующий параметр, дисковая активность снизится на треть:

```
set fast-update              = false
```

Однако данный параметр не сочетается с локальными почтовыми клиентами. Кроме того, нарушится работа программ UNIX, извещающих о приходе новой почты. Такое значение можно безопасно применять на почтовом сервере, где клиенты забирают почту только по POP.

Настройка APOP

APOP применяет систему «оклик/отзыв» (challenge-response system). Когда пользователь подключается к POP3-серверу, сервер берет известный пароль пользователя и вычисляет на его основе «оклик». «Оклик» посылается клиенту. Клиент принимает «оклик», комбинирует его с паролем и посылает обратно на сервер. Тем временем сервер выполняет те же вычисления и сравнивает «отзыв» клиента со своим результатом. Если они соответствуют, значит, клиент подтвердил наличие пароля. Загрузка почты разрешена.

Зачем все это? Затем, что сам пароль никогда не передается по сети. Следовательно, при запросах к qpopper ваш пароль украсть никому не удастся. Однако пароль может быть раскрыт при работе с веб-броузером, telnet и т. д.

База паролей APOP

Поскольку APOP вычисляет «разделяемое секретное значение» на основе пароля пользователя, программе qpopper нужен доступ к паролям. В UNIX шифрование паролей – это дорога в один конец; даже получив доступ к файлу

/etc/master.passwd, вы не сможете извлечь пароль.[1] Поэтому для реализации APOP необходима отдельная база данных с именами пользователей и паролями. Такая база хранится в */usr/local/etc/qpopper/pop.auth.db*. Доступ на чтение этого файла должен быть только у root.

Администрировать базу пользователей APOP можно с помощью qpopauth(8). Прежде всего, базу надо инициализировать:

```
# qpopauth -init
```

Теперь, когда есть база, можно задействовать qpopauth для управления пользователями.

Добавление пользователей

Следующая команда добавляет пользователя в базу:

```
# qpopauth -user username
```

Вам будет предложено ввести пароль. Если пользователь в основной системе не существует, qpopauth не позволит добавить его в базу.

Удаление пользователей

Следующая команда удалит указанного пользователя из базы данных:

```
# qpopauth -delete username
```

Просмотр списка пользователей

Пользователь, запустивший следующую команду, узнает, есть ли он в базе пользователей APOP. Если root выполнит `qpopauth -list ALL`, будет выведен список всех пользователей APOP:

```
# qpopauth -list
```

Включение APOP

При настройке APOP надо решить, будет ли разрешена передача по POP3 незашифрованного текста. Для того чтобы разрешить простой POP3 и APOP, надо изменить значение параметра `clear-text-password`. (Значение, определенное по умолчанию, позволяет передавать пароли открытым текстом лишь в тех случаях, когда клиентские программы не поддерживают APOP.)

Параметр `clear-text-password`, задаваемый в конфигурационном файле qpopper, определяет необходимость применения APOP:

```
set clear-text-password = always
```

Поддержка APOP

Разрешение как APOP, так и простого POP3 может привести к путанице с паролями, поскольку база пользователей APOP и файл */etc/master.passwd*

---

[1] Можно предпринять так называемую «атаку грубой силой» и попытаться найти текстовую строку, зашифрованный вариант которой совпадет с зашифрованным паролем. Такой перебор подразумевает значительную загрузку процессора, требует времени и совершенно не подходит для серверного протокола.

могут быть синхронизированы только благодаря прямому вмешательству администратора. Когда пользователь звонит и говорит, что не может забрать почту, надо выяснить, какой протокол он применяет: APOP или POP3. Обычно пользователь об этом не знает, поэтому вам придется задавать вопросы по поводу почтового клиента или просто изменить оба пароля, задав известное значение. Выбор APOP предпочтительнее со всех точек зрения.

А еще лучше выбрать pop3ssl.

Конфигурирование pop3ssl

Процесс POP3-через-SSL похож на протокол POP3 по умолчанию. Однако вместо отправления имени пользователя клиент посылает запрос на применение SSL. Если сервер предоставляет такую возможность, все остальные данные будут шифроваться.

Все параметры, относящиеся к производительности, устанавливаются так же, как и для стандартного POP3. Однако в случае pop3ssl необходимо установить несколько дополнительных параметров.

```
set clear-text-password = tls
```

С помощью параметра `clear-text-password` можно передавать пароли открытым текстом в случае применения шифрования SSL. Пользователь может применять APOP или pop3ssl, но не простой POP3.

```
set tls-cipher-list = bf,des,des3
```

Параметр `tls-cipher-list` перечисляет криптографические шифры, поддерживаемые вашей системой. Полный список шифров, поддерживаемых системой, можно получить с помощью команды `openssl list-cipher-commands`. Шифры из предыдущего примера поддерживаются большинством почтовых клиентов.

```
set tls-server-cert-file     = /usr/local/etc/qpopper/server.cert
```

Параметр `tls-server-cert-file` задает месторасположение файла с подписанным сертификатом. Получение подписанного серверного сертификата было рассмотрено в предыдущей главе.

После установки этих параметров все готово для предоставления сервиса pop3ssl. Безусловно, это самый предпочтительный метод, реализовать который довольно легко.

Безопасность qpopper

qpopper имеет сомнительную репутацию с точки зрения безопасности, однако его код подвергся значительной переработке. Теперь qpopper так же защищен, как и любой другой демон POP3. Однако вам по-прежнему надо интересоваться вопросами безопасности qpopper, как в случае любой программы, которая передает данные по сети. Поскольку qpopper запускается из inetd, для его защиты можно применить TCP Wrappers.

15

Веб- и FTP-сервисы

Интернет возник в 1970-х годах, но по-настоящему популярным стал в середине 1990-х годов с появлением веб-страниц. Корпорация Netscape взяла веб-броузер Mosaic, распространяемый с открытым исходным кодом, и преобразовала его в коммерческий продукт. Это событие стало началом не прекращающейся и сейчас революции в отображении информации и обмене ею. Многие высокотехнологичные компании появлялись на свет и исчезали, однако эра взаимодействия «человек–человек» (person-to-person) началась с появлением веб-броузера Netscape. Такие технологии, как peer-to-peer, расширили Интернет еще больше, однако Сеть (Web) – это первое, что ассоциируется со словом «Интернет».

О производительности веб-сервера FreeBSD ходят легенды. Многие годы даже подразделения Microsoft отдавали предпочтение FreeBSD, а не собственной платформе Windows NT. (В тот самый день, когда я писал эти строки, газета «Wall Street Journal» сообщила, что Microsoft по-прежнему применяет FreeBSD в своих стенах, хотя представители корпорации утверждают обратное.) Это вылилось в создание комплекта инструментов «.NET для FreeBSD», выпущенного Microsoft в виде условно бесплатного программного обеспечения. Система FreeBSD работает в Yahoo!, а также в великом множестве других крупных веб-компаний, предъявляющих высокие требования к платформе. Веб-сервер Apache, самый популярный веб-сервер в мире, разрабатывается на системе FreeBSD.

В этой главе обсуждается создание высокопроизводительного веб- и FTP-сервера на базе FreeBSD.

Как работает веб-сервер

Алгоритм работы веб-сервера довольно прямолинеен: веб-броузер запрашивает страницу, веб-сервер выдает ее и закрывает соединение. Это довольно просто. Дело значительно усложняется, когда вы начинаете применять модули, динамические страницы и т. д., однако в данной главе рассмотрены только основы.

В Сети применяется протокол передачи гипертекста (Hypertext Transfer Protocol, HTTP) – очень простой протокол, как и POP3. За последние несколько лет в HTTP добавлены функции, которые сделали его более сложным, однако базовые операции HTTP настолько просты, что их можно выполнять вручную. Давайте попробуем. С помощью telnet(1) подключимся к серверному порту 80 и наберем команду `GET /`.

```
# telnet blackhelicopters.org 80
Trying 209.69.178.18...
Connected to blackhelicopters.org.
Escape character is '^]'.
GET /
<font color=white> Nothing to see here.

This is not the site you're looking for.
Connection closed by foreign host.
#
```

Если вы когда-нибудь видели HTML-страницу, то вывод этой команды покажется вам очень знакомым. Если нет, то когда в следующий раз загрузите веб-страницу, выполните в своем броузере команду View Source (Просмотр исходного кода) или аналогичную. Тогда вы увидите, что привлекательная страница, появляющаяся в броузере, создается средствами HTML. (Если страницы не загружаются с веб-сервера, в нем могут быть неполадки. Проверьте протоколы регистрации ошибок.)

В состав FreeBSD входят несколько веб-серверов, однако самым популярным в Интернете, безусловно, является Apache.

Веб-сервер Apache

Заглянув в каталог */usr/ports/www*, можно найти несколько различных «портов», в именах которых фигурирует «apache». Большая их часть представляет собой версии веб-сервера Apache, слегка отличающиеся друг от друга. Следовательно, перед установкой надо выбрать версию. Вот краткий обзор вариантов выбора.

apache-jserv

Это веб-сервер Apache с машиной сервлетов Java. Его можно применять для обслуживания JavaServer Pages.

apache13

Может быть, эта версия вам понравится: Apache 1.3 без дополнительных функций. Тем не менее при установке Apache могут потребоваться крупные или далеко идущие изменения в самом Apache. Изучите другие «порты» Apache, – возможно, они подойдут лучше.

apache13+ipv6

Это Apache с изменениями для поддержки соединений IPv6. Выберите данную версию, если работаете с IPv6.

apache13-fp

Microsoft FrontPage Server Extensions – это популярная платформа для разработки веб-приложений, но ее установка может оказаться делом хлопотным. Выберите данный «порт», если необходима поддержка FrontPage.

apache13-modssl

В данный «порт» включены модули поддержки протокола защищенных сокетов (Secure Sockets Layer, SSL), предназначенные для защиты веб-сайтов. Компонент защиты соединения, mod_ssl, основан на OpenSSL. Применяйте этот «порт» для поддержки защищенных соединений.

apache13-ssl

В состав «порта» интегрированы средства (немодульные) поддержки SSL, которые считаются устаревшими; вместо них рекомендуется применять mod_ssl.

apache2

Это не просто передний край – это последнее слово в веб-серверах. Данная версия может выжать из вас все соки. apache 2 заслуживает установки хотя бы потому, что поможет идти в ногу с технологиями, однако пока его не стоит применять в реальных задачах. Кроме того, многие модули Apache еще не перенесены в apache2. Если вам нужен легкий базовый веб-сервер, в котором реализованы самые современные технологии, эта версия будет в самый раз.

Наиболее эффективная установка Apache подразумевает применение «портов». Процесс сборки займет больше времени, но в результате будет собран более устойчивый, быстрый веб-сервер. Для достижения наилучшей производительности при сборке из выбранного «порта» следует задать `APACHE_PERF_TUNING=YES`:

```
# make APACHE_PERF_TUNING=YES all install
```

Конфигурационные файлы Apache

Основные конфигурационные файлы Apache расположены в */usr/local/etc/apache*. Есть пять основных файлов: *access.conf*, *httpd.conf*, *magic*, *mime.types* и *srm.conf*. Первоначально широко применялись все пять файлов, а в настоящее время чаще используются *httpd.conf*, *magic* и *mime.types*. (Функции *access.conf* и *srm.conf* были перенесены в *httpd.conf*; первоначальные файлы оставлены в основном для умудренных опытом администраторов, которые привыкли к их наличию.)

Для того чтобы администрировать Apache должным образом, необходимо понимать назначение каждого файла и способы управления ими.

mime.types

Файл *mime.types* содержит список типов стандартных файлов и их идентифицирующие характеристики. Все веб-серверы должны идентифицировать

тип файла, который они передают клиенту, чтобы тот мог выполнить соответствующие действия. Например, при загрузке файла PDF большинство веб-броузеров открывают Acrobat Reader. Соответствия, хранимые в *mime.types*, предоставляют серверу Apache информацию, необходимую для поддержки его функциональности. Вносить изменения в данный файл следует лишь при наличии прямых указаний в документации той или иной программы.

magic

В этом файле хранятся идентифицирующие характеристики необычных типов файлов, не вошедших в файл *mime.types*. Поскольку файл *mime.types* не может учитывать все возможные типы, модуль mime_magic, встроенный в Apache, применяет файл *magic* для идентификации неизвестных файлов. Вносить изменения в данный файл следует лишь при наличии прямых указаний в документации той или иной программы.

httpd.conf

Файл *httpd.conf* управляет работой веб-сервера и представляет особый интерес. Этот файл снабжен подробными комментариями (все строки, начинающиеся со знака «решетки»), поэтому здесь он не будет рассматриваться подробно, а обсуждение всех возможных вариантов конфигурации Apache мы оставим для более толстых книг. Однако в данном файле размещены большинство конфигурационных параметров Apache, поэтому мы не можем обойти его стороной.

Конфигурирование Apache

Конфигурационный файл Apache лучше всего создать путем редактирования файла, поставляемого в качестве исходного примера (*httpd.conf*). Однако, как бы вы ни поступили, не изменяйте параметры, назначение которых вам непонятно. В отличие от сервера DNS, конфигурационный файл Apache не надо создавать с нуля! По умолчанию файл *httpd.conf* содержит большие разделы параметров, которые управляют, например, кодировками символов. Если ваш веб-сервер не должен обслуживать страницы на китайском языке и отличаться от всех других веб-серверов на планете, лучше всего оставить значения по умолчанию.

Примечание

Размещение параметров в файле *httpd.conf* немного хаотично. Такая компоновка имеет определенный смысл с точки зрения авторов, но нам остается только недоумевать, разбираясь с его содержимым. (По умолчанию файл содержит более 1000 строк, и это делу не помогает!) Мы же приступим к рассмотрению параметров настройки в более разумной последовательности.

Общие настройки сервера

Следующие параметры настройки определяют работу сервера в целом.

Путь к корню сервера

Параметр ServerRoot определяет путь к файлам основного веб-сайта.

```
ServerRoot "/usr/local"
```

Если ваш сервер обслуживает один большой сайт, то его можно разместить в другом месте – в разделе, предназначенном для веб-файлов.

Примечание

Если ссылка на тот или иной файл в *httpd.conf* не предваряется косой чертой (/), Apache присоединяет спереди значение ServerRoot. Например, если ServerRoot представлен как /usr/local, путь "docs/cgi-bin" превратится в "/usr/local/docs/cgi-bin", а "/usr/local/etc/apache/vhost.conf" не изменится.

Путь к корню документов

```
DocumentRoot "/usr/local/www/data"
```

Документы HTML для основного веб-сайта размещаются в каталоге, указанном в DocumentRoot.

Серверы httpd

```
MinSpareServers 5
MaxSpareServers 10
StartServers 5
```

Если на сервере выполнить команду ps -ax, можно увидеть некоторое множество процессов httpd, поскольку каждый запрос к веб-серверу обслуживается отдельным процессом. Когда десяток пользователей набирают URL вашего веб-сервера и одновременно нажимают Enter, для обслуживания каждого запроса создается отдельный процесс. Отчасти такой алгоритм объясняет, как Apache управляется с такой высокой нагрузкой.

Когда Apache запускается, он создает процессы httpd, количество которых равно значению StartServers. Время от времени он проверяет общее количество запущенных процессов httpd, а также количество процессов, которые действительно заняты обслуживанием содержимого. Для того чтобы количество процессов httpd было достаточным для обслуживания дополнительных запросов, Apache применяет значения MinSpareServers и MaxSpareServers.[1]

Если на веб-сервер обрушивается необжиданно большой объем трафика, значения MinSpareServers и MaxSpareServers можно увеличить. Однако значе-

[1] Если запросов больше, чем запущенных при старте копий Apache (количество последних задано параметром StartServers), то количество копий увеличивается для обслуживания запросов. Эти дополнительные процессы не завершаются после ответа на запрос, а остаются в памяти. Директива MaxSpareServers задает количество запущенных незанятых копий. Если их больше, чем указано в директиве MaxSpareServers, то лишние процессы завершаются. Аналогично, если свободных серверов в пуле меньше, чем предписано параметром MinSpareServers, то в преддверии наплыва запросов создаются дополнительные копии сервера. – *Примеч. науч. ред.*

ние StartServers увеличивать не надо: даже если необходимо остановить и перезапустить Apache, он может обработать несколько сотен процессов httpd в течение нескольких секунд.

Максимальное количество клиентов

```
MaxClients 150
```

Параметр MaxClients определяет максимальное количество процессов httpd, которые Apache имеет право запустить одновременно. Если предел достигнут, веб-сервер может стать недоступным. Такое ограничение предназначено для старых систем; значение этого параметра на большинстве современных компьютеров можно легко увеличить. Количество процессов httpd, необходимых системе, можно выяснить экспериментальным путем.

Примечание

Количество запущенных процессов httpd в любое время можно узнать с помощью команды ps -ax | grep httpd | wc -l.

Максимальное количество запросов httpd

Каждый запущенный процесс httpd обрабатывает тот или иной запрос, а затем ждет поступления следующего. Если на сервере работает веб-программа, которая приводит к утечке памяти, вынуждая каждый процесс httpd использовать все больше и больше памяти, то увеличение размера каждого процесса httpd позволит обнаружить команда top.

Если если на вашей системе имеют место такие трудности с использованием памяти, то с помощью параметра MaxRequestsPerChild можно задать останов процесса после того, как он обслужит определенное количество запросов. Значение 0 подразумевает, что каждый процесс может обслуживать неограниченное количество запросов:

```
MaxRequestsPerChild 0
```

Большинство систем FreeBSD отлично работают при значении 0, однако его можно изменить, если процессы httpd потребляют много памяти. Как правило, в этом случае источником неприятностей оказывается какое-нибудь веб-приложение.

Listen

```
Listen 80
```

Параметр Listen определяет, к каким TCP-портам или IP-адресам будет привязан Apache. Можно указать отдельные IP-адреса:

```
Listen 192.168.8.44
```

Далее объединим адрес и номер порта, чтобы запустить Apache на необычном порту:

```
Listen 192.168.8.44:88
```

Или можно прослушивать необычный порт на всех IP-адресах системы:

```
Listen 88
```

Можно задать несколько операторов Listen, обеспечив доступность сервера Apache на любом порту или IP-адресе системы.

BindAddress

```
BindAddress *
```

Подобно оператору Listen, BindAddress указывает IP-адрес, на котором Apache будет ожидать соединений. По умолчанию Apache подсоединяется к каждому сетевому интерфейсу системы, однако с помощью данной опции можно разрешить только один IP-адрес. По существу, BindAddress идентичен оператору Listen.

Модули

```
LoadModule
AddModule
```

С помощью модулей можно добавлять в Apache дополнительные функции. Модули, перечисленные в базовой конфигурации, предоставляют основную функциональность Apache, поэтому не изменяйте существующие операторы LoadModule и AddModule без веских причин. (Модули Apache рассматриваются более подробно в разделе «Модули Apache» этой главы.)

Порт

```
Port 80
```

Это TCP-порт, который прослушивается сервером Apache. Можно применять несколько операторов Port.

Пользователь и группа

```
User www
Group www
```

Эти опции задают пользователя и группу системы UNIX, под которыми запускается Apache. Когда вы входите в систему FreeBSD и запускаете ту или иную программу, она получает ваши права доступа. Точно так же веб-сервер Apache запускается от имени определенного пользователя и получает его права доступа.

В последних системах FreeBSD предусмотрены пользователь www и группа www. Это специальные учетные записи без привилегий, предназначенные для веб-сервера. (Вы не можете войти в систему под именем www.) Хотя иногда встречаются рекомендации запускать Apache от имени root, не поступайте так ни по каким причинам. Если злоумышленник взломает ваш веб-сервер, в качестве дополнительной награды он получит права root в вашей системе!

Почтовый адрес администратора

```
ServerAdmin webmaster@AbsoluteBSD.com
```

Этот параметр задает почтовый адрес специалиста, запустившего сервер. Если кто-нибудь заметит неполадки на сервере, он может послать письмо по данному адресу.

Имя сервера

```
ServerName www.AbsoluteBSD.com
```

Это имя веб-сайта. Оно должно представлять собой реальное имя хоста и иметь соответствующую запись в DNS. В противном случае запустить Apache не удастся. Однако во время тестирования вместо записи DNS можно применять запись в */etc/hosts*.

Индекс каталога

Оператор `DirectoryIndex` задает имя документа, выдаваемого по умолчанию. Когда клиент обращается к каталогу, а не к имени файла, Apache ищет в этом каталоге файлы с указанными именами согласно порядку их представления в данном операторе.

```
DirectoryIndex index.php index.html
```

Если веб-броузер запрашивает *http://www.AbsoluteBSD.com/refunds/*, Apache ищет *index.php*, затем *index.html* и возвращает первый попавшийся файл из указанных в `DirectoryIndex`.

По умолчанию в файле *httpd.conf* заложен набор довольно сложных правил для определения значения `DirectoryIndex`. Это отличный пример условного параметра, значение которого определяется в зависимости от загруженных модулей. Если вы не используете дополнительные модули, то можете убрать все правила и заменить их на простую однострочную запись.

Поиск имен хостов для протоколов

```
HostnameLookups Off
```

Этот параметр устанавливает, какую информацию сохраняет Apache – IP-адреса или имена хостов. Если сохранять имена хостов, то протоколы будут выглядеть лучше – место IP-адресов займут имена компьютеров, однако эту возможность, как правило, лучше отключить. Поиск имен добавит нагрузку на сервер, а большинство программ, анализирующих протоколы, выполняют такой поиск самостоятельно.

Местонахождение протокола регистрации ошибок

```
ErrorLog /var/log/httpd-error.log
```

Здесь находится протокол регистрации ошибок.

Уровень протоколирования

```
LogLevel warn
```

Оператор LogLevel управляет объемом протоколируемых данных. Действительные значения: debug, info, notice, warn, error, crit, alert и emerg. По этим значениям системный регистратор (system logger) определяет объем протоколируемых данных. Значение warn представляется разумным.

Формат протокола

```
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
```

Оператор LogFormat определяет, какие данные будут протоколироваться. Для удобства настройки предусмотрено несколько предопределеннных вариантов LogFormat:

- Формат «common» подразумевает протоколирование IP-адреса клиента, времени запроса, имени запрашиваемого файла и некоторой другой базовой информации.
- Формат «referrer» подразумевает протоколирование сайта, с которого клиент перешел на данный сайт. Другими словами, запоминается другая веб-страница, на которой есть ссылка на вашу страницу.
- Формат «agent» предполагает запись информации о веб-броузерах клиентов.
- Формат «combined» подразумевает все вышеперечисленное. (Выбор данного формата может привести к очень большим протоколам, поэтому необходимо обеспечить достаточное дисковое пространство.)

Примечание

Если покопаться в документации Apache, можно довольно легко написать собственные операторы LogFormat. Однако представленные здесь форматы распознаются всеми популярными программами для анализа веб-протоколов, поэтому не изменяйте их без достаточных оснований.

Custom Log

```
CustomLog /var/log/httpd-access.log combined
```

Оператор CustomLog задает имя файла протокола и применяемый формат. В предыдущем примере обращения к основному веб-сайту протоколируются в */var/log/httpd-access.log* в формате «combined».

Для различных целей можно применять несколько операторов CustomLog. Допустим, вы применяете модуль Apache, предоставляющий особые функции, которые вы хотите протоколировать. В то же время вы не хотите мешать работе программы WebTrends, применяемой вашими клиентами. В таком случае с помощью двух операторов CustomLog можно осуществлять запись в два отдельных файла протокола.

Псевдонимы

```
Alias /icons/ "/usr/local/www/icons/"
```

Оператор `Alias` задает псевдонимы для каталогов на вашем веб-сайте – подобно символическим ссылкам. В предыдущем примере пользователь, заглянувший на *http://www.AbsoluteBSD.com/icons/*, на самом деле получает файлы из каталога */usr/local/www/icons/*.

Ошибка при доставке документа

```
ErrorDocument 404 /missing.html
```

Когда броузер запрашивает документ, отсутствующий на сервере, сервер возвращает сообщение об ошибке. С помощью директивы `ErrorDocument` можно указать страницу, возвращаемую при возникновении ошибки.

Include

```
Include /usr/local/etc/apache/vhosts.conf
```

Файл *httpd.conf* может включать в себя другие файлы. Такая возможность полезна при наличии нескольких подсистем, поддерживаемых различными людьми. Кроме того, это позволяет лучше систематизировать конфигурацию.

Например, многие дополнительные модули Apache имеют свои параметры настройки. Проще предусмотреть для каждого модуля свой конфигурационный файл, чем сваливать все параметры в основной файл *httpd.conf*. Аналогично, если тот или иной сотрудник отвечает за поддержку виртуальных хостов, можно создать файл конфигурации виртуальных хостов и предоставить к нему доступ этому сотруднику (или группе сотрудников) (см. главу 8).

Конфигурирование раздела Directory

В Apache есть много отличных функций, однако не стоит включать их всегда и везде – неряшливая настройка может привести к выдаче слишком большого объема информации с вашего веб-сайта и даже к взлому сервера.

Права на доступ к функциям Apache предоставляются по каталогам. Конфигурация очень похожа на XML: есть метка `Directory` в угловых скобках, список прав доступа и параметров, а затем закрывающая запись `Directory` с обратной косой чертой. Любые значения или параметры между открывающим и закрывающим операторами `Directory` оказывают влияние на этот каталог. Вот базовый формат:

```
<Directory /путь/к/файлам>

...здесь расположены параметры и значения

</Directory>
```

По умолчанию Apache применяет весьма ограничивающие права и значения параметров. Например, в файле *httpd.conf* в начале раздела, описывающего каталоги, можно увидеть следующую запись:

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
```

Apache позволяет пользователям загружать на сервер свои конфигурационные файлы, которые изменяют работу сервера. Коротко говоря, пользователи могут задать свои опции, защиту паролей, типы MIME и т. д. Строка `AllowOverride None`, показанная в предыдущем примере, означает, что пользователи не смогут установить те или иные опции в указанном каталоге.

И еще: косая черта (/) представляет собой каталог ServerRoot, который указан в общих настройках сервера. Если иное не оговорено, каждый каталог на сервере имеет такие же права доступа, однако в отдельных каталогах эти значения можно подменить. (Опции прав доступа будут кратко обсуждаться в разделе «Конфигурирование раздела Directory» далее в этой главе; сейчас надо просто понять, что никто не сможет жульничать без нашего разрешения.) Чтобы убрать права доступа, предоставляемые по каталогам, обратитесь к исходному файлу *httpd.conf*. В нем можно найти минимальный[1] набор прав для различных каталогов.

Функции, указываемые в разделе Directory

Теперь, когда мы запретили все функции, которые предоставляет Apache, можно явно указать необходимые нам функции. Это позволит вашим веб-дизайнерам выполнять много различных операций, но вам, как администратору, должно быть известно назначение каждой операции. Рассмотрим некоторые функции, которые можно задать для каталога.

Управление доступом с помощью IP-адресов и сетевых блоков

Параметры `Allow` и `Deny` задают IP-адреса и имена хостов, с которых разрешен доступ к содержимому каталога. Адрес клиента сравнивается со списками «allow» и «deny» в порядке, указанном в операторе `Order`. Далее в зависимости от результата сравнения Apache разрешает или запрещает доступ к файлам каталога. Если порядок `Order` определен как `deny,allow`, то по умолчанию доступ открыт, если нет запрещающего оператора `Deny`. Если же порядок определен как `allow,deny`, то по умолчанию доступ запрещен, если нет разрешающего оператора `Allow`.

Разрешение или запрет доступа с тех или иных хостов задается с помощью IP-адресов и имен хостов. Каждый запрос на соединение интерпретируется в соответствии с указанными правилами. Каждый запрос на соединение является частью специальной группы *All*. Подобно группам в TCP Wrappers, группу All и дополнительную идентификацию клиента можно применять для разрешения или запрета доступа с определенных хостов.

Запутались? Рассмотрим пример.

```
Order allow,deny
Allow from all
Deny from *.AbsoluteBSD.com
```

[1] Для детальной настройки вам все равно придется читать документацию по Apache. При установке Apache из исходных текстов документация расположена в подкаталоге *htdocs* того каталога, куда вы установили веб-сервер. – *Примеч. науч. ред.*

Я захожу на сайт с машины *AbsoluteBSD.com*. Первым делом Apache проверяет список `Order` (поскольку это первый оператор, с которым он столкнется). Согласно `Order`, Apache должен обратиться к списку `Allow`, а затем к списку `Deny`. Поскольку *AbsoluteBSD.com* входит в `all`, то сначала мне разрешается доступ. Однако затем Apache проверяет список `deny`. Приехали. Я отброшен.

Заметим, что Apache берет имена хостов из обратного DNS. Поскольку в большинстве случаев администратор может легко изменить имя хоста в записи обратного DNS, при управлении доступом безопаснее применять IP-адреса или даже блоки IP-адресов.

В следующих строках представлены допустимые типы IP-адресов и аргументы блоков IP-адресов для операторов `Allow` и `Deny`:

```
192.168.0.1,192.168.0.2,192.168.0.3
192.168
192.168.0.0/16
192.168.0.0/255.255.0.0
```

Первая строка управляет тремя IP-адресами. Последние три строки означают одно и то же – это просто три разных способа представления всех IP-адресов, начинающихся с 192.168.

Доступ к внутреннему веб-сайту можно разрешить только с хостов вашей корпоративной сети. Для этого достаточно задать такие строки:

```
Order allow,deny
Allow from 192.168.1/24
Deny from all
```

(Безусловно, такая конфигурация предполагает, что корпоративный брандмауэр блокирует поддельный трафик.)

Параметры httpd.conf

Параметры представляют основные серверные функции, которые можно включать или выключать для отдельных каталогов. Они позволяют веб-разработчику выполнять программы на сервере, включать и выключать защиту с помощью паролей для тех или иных каталогов и управлять поддержкой языков. Параметры предоставляют веб-разработчику мощные возможности. Они могут вызвать дополнительные вопросы, поэтому необходимо понимать назначение и действие каждой из них.

Параметры для каталогов можно указывать с помощью ключевого слова `Options`. Например, чтобы задать параметр `ExecCGI` для каталога */usr/local/www/data/catalog*, надо выбрать такую конфигурацию:

```
<directory /usr/local/www/data/catalog>
     Options ExecCGI
</directory /usr/local/www/data/catalog>
```

Кроме того, в одной строке можно указать несколько параметров:

```
<directory /usr/local/www/data/catalog>
     Options ExecCGI,MultiViews
</directory /usr/local/www/data/catalog>
```

Теперь изучим стандартные параметры.

`All`

Параметр `All` включен в сервере Apache по умолчанию. Если файл *httpd.conf* пуст, в каталогах будут доступны почти все функции Apache. Если пользователь загрузит на сервер сценарий для защиты своего каталога с помощью паролей, это сработает. Пользователь может загрузить на сервер CGI-сценарий, который пробьет брешь в локальной системе и запустит командный процессор от имени root на TCP-порту, открыв всем желающим двери в вашу систему. Параметр `All` разрешает все остальные параметры Apache, за исключением `MultiViews` (будет кратко описан далее).

В файле *httpd.conf*, поставляемом с Apache в качестве примера, для каталога ServerRoot намеренно задано `Options None`. Это означает отключение разрешительных правил Apache, заданных по умолчанию. Если в качестве основы для *httpd.conf* вы применяете файл-образец (как и следует поступать), параметр `All` будет отключен. Следовательно, вам придется специально включать параметры в тех каталогах, где вы хотите применять их.

`ExecCGI`

Если этот параметр для каталога задан, то можно запускать CGI-сценарии, находящиеся в этом каталоге.

`FollowSymLinks`

Для обращения к другим файлам на сервере можно применять символические ссылки (или псевдонимы, описываемые в ln(1)). Пользователь может создать символическую ссылку чуть ли не на каждый файл на сервере, причем этот файл будет видимый (конечно, если права доступа к файлу разрешают это).

`Includes`

Вставки на стороне сервера (Server Side Includes, SSI-файлы HTML, содержащие инструкции для командного процессора) и CGI-сценарии будут работать в каталогах с установленным параметром `Includes`, однако они могут быть опасны, если их писали неосторожно. В конце концов, вы позволяете посетителям вашего веб-сайта запускать команду, которая использована в этом HTML-файле. Коварные взломщики, проявив изобретательность, могут с помощью данной команды выполнить действия, которые не предусматривал веб-дизайнер. (Поищите в Сети материалы, посвященные SSI и безопасности, и вы найдете описание различных ловушек.) Если вы не знаете, как безопасно применять SSI, не включайте эту опцию!

`IncludesNOEXEC`

Разрешает SSI, но запрещает функцию `#exec SSI` и функцию `include` в CGI-сценариях. Без функции `#exec` код HTML не может просто запустить любую команду – все команды SSI должны быть написаны в тщательно ограниченном диапазоне. По сути, этот параметр разрешает простые вставки SSI и сценарии CGI, но исключает наиболее типичные дыры в безопас-

ности. Зайдите на Google и поищите описания различных проблем безопасности, обусловленных небрежно написанными вставками SSI.

`Indexes`

Если в каталоге нет файла *index.html* или аналогичного (такого, который указан в `DirectoryIndex`), сервер вернет хорошо отформатированный список содержимого каталога. Можно считать это брешью в системе безопасности – в зависимости от содержимого. Например, если кто-либо просматривает содержимое каталога с моей домашней веб-страницей, это не страшно. Но если открыт доступ к каталогу с секретным кодом, дело принимает другой оборот.

`MultiViews`

Сервер может обслуживать документы, представленные на нескольких языках. Например, веб-разработчик может создать документ HTML, содержащий текст на английском, китайском и испанском языках. Если параметр `MultiViews` включен, Apache посылает клиенту страницу на языке, который задан в веб-броузере.

`None`

Параметры не разрешены. Каждая из замечательных возможностей, описанных здесь, не будет работать.

`SymLinksIfOwnerMatch`

Сервер будет применять символические ссылки, если владелец ссылки также владеет файлом, на который указывает ссылка. Другими словами, пользователь может применять символические ссылки, указывающие на его документы.

`AllowOverride`

Если разрешить подмену, пользователи смогут изменять опции, разрешенные в том или ином каталоге. Apache проверяет каждый каталог на наличие файла *.htaccess* и воспринимает содержимое данного файла как дополнительную информацию о конфигурации сервера. Такая возможность позволяет веб-разработчикам самим определять типовые значения параметров и устанавливать ненадежные сценарии CGI в произвольных каталогах.

Для подмены значений параметров надо разрешить или запретить применение *.htaccess*. Если вы запускаете корпоративный веб-сервер, а ваш веб-разработчик так или иначе получает все, что ему нужно, нет никаких причин запрещать ему подмены. Но если вы запускаете общедоступный веб-сервер или веб-сервер компании-провайдера и хотите запретить определенным группам клиентов выполнять сценарии CGI, то надо запретить те или иные подмены.

Для параметра `AllowOverride` определены следующие операторы, разрешающие пользователям подменять что угодно с помощью записи в *.htaccess*.

- `AuthConfig` позволяет защищать каталоги с помощью паролей и обеспечивает довольно высокую степень безопасности. Ее применение имеет

смысл при администрировании ферм серверов, где учетную запись может получить любой, у кого есть кредитная карта.

- `FileInfo` разрешает пользователям вставлять собственную информацию MIME для каталога. Как правило, лучше добавлять такую информацию в серверный файл *mime.types*, но такая возможность может потребоваться, например, когда пользователи загружают на сервер файл, который применяется только в их компании, при этом требуется, чтобы броузеры воспринимали его должным образом.
- `Indexes` разрешают управление индексами, а именно задание имени файла, выдаваемого по умолчанию, настройку представления пиктограмм в индексах, созданных сервером, и т. д.
- `Limit` разрешает применение ключевых слов `Allow`, `Deny` и `Order`. Эта опция также вполне безопасна.
- `None` означает, что пользователь ничего не может изменять. Это хороший выбор по умолчанию, однако он налагает излишние ограничения на работу большинства приложений.
- `Options` разрешает установку любых опций, описанных здесь. Она полезна, если вы знаете веб-разработчиков и доверяете им либо не озабочены тем, что пользователь может загрузить небезопасную программу, подвергающую сервер риску.

С учетом вышесказанного разумный набор значений по умолчанию может выглядеть так:

```
AllowOverride Limit, AuthConfig, Indexes, FileInfo
```

Управление HTTP-запросами

Метод HTTP – это команда, посылаемая броузером серверу. Вероятно, вы слышали о таких методах HTTP, как GET и POST. Они предписывают серверу передавать определенную страницу и обрабатывать загружаемую информацию соответственно. Однако из методов, каждый из которых имеет свое предназначение, можно составить целый список. Ограничить количество методов, принимаемых сервером, позволяет директива `Limit`.

Директивы `Limit` и `LimitExcept` задают методы HTTP, которые можно применять в каталоге. В большинстве случаев забота о методах, применяемых для той или иной веб-страницы, ложится на плечи веб-разработчика. Однако вы должны уметь разрешать или запрещать такие методы. (Если вы – веб-разработчик, читающий эту книгу для изучения работы сервера, тем лучше для вас! Теперь поищите описание методов в книге по HTTP.)

Директива `Limit` явно перечисляет методы, применение которых разрешено в данном каталоге, а также правила их применения в формате «allow» и «deny». Директива `Limit` была рассмотрена в предыдущем разделе «Параметры httpd.conf», когда обсуждалось ограничение доступа к каталогу с помощью указания определенных IP-адресов.

Директива `LimitExcept` похожа на `Limit` за исключением того, что создаваемые правила применимы только к необъявленным методам. Подобно дирек-

тиве Directory, в Limit и LimitExcept для указания начала и конца блока применяются угловые скобки < и >.

Например, чтобы запретить всем пользователям загрузку файлов на сервер, можно применить следующую конфигурацию:

```
<Directory />
  <Limit GET POST OPTIONS PROPFIND>
    Order allow,deny
    Allow from all
  </Limit>
  <LimitExcept GET POST OPTIONS PROPFIND>
    Order deny,allow
    Deny from all
  </LimitExcept>
</Directory>
```

В первой части приведенного кода, начиная с Limit, создается список методов HTTP, которыми мы хотим управлять: GET, POST, OPTIONS и PROPFIND. Как мы уже говорили, директивы Order и Allow разрешают применение любого метода из списка. Раздел завершается директивой /Limit.

В разделе LimitExcept создается список значений, не включенных в список Limit. (Это такой же список, как и в первом разделе.) Мы не хотим блокировать значения, указанные в предыдущем списке, но мы хотим блокировать значения, которые не представлены в данном списке. Вот почему мы применяем LimitExcept. Здесь директивы Order и Deny запрещают применение методов, не указанных в данном списке.

Если вы добавляете метод HTTP (скажем, из модуля Apache), модуль LimitExcept добавит его автоматически.

Директивы Limit и LimitExcept можно применять и для других целей, однако при наличии интереса к этой теме лучше обратиться к одной из толстых книг о сервере Apache.

Каталоги, защищаемые с помощью паролей

Поговорим о защите каталогов веб-сайта с помощью паролей. Такую защиту нетрудно организовать с помощью текстового файла, содержащего имена пользователей и зашифрованные пароли, однако, как и другие легкие и простые методы, этот характеризуется довольно низкой скоростью. А мы создаем высокопроизводительный веб-сервер и поэтому не ищем легких путей.[1] Мы выберем более трудную дорогу, которая приведет к лучшим рельзутатам.

Подобно тому как имена и пароли пользователей системы хранятся в базе данных для эффективного доступа к ним, Apache может применять базу данных для хранения своих имен пользователей и паролей. Для аутентифи-

[1] Каждый раз, когда я применял метод на скорую руку, мне приходилось спустя месяцы или годы возвращаться назад и применять более правильный метод. Выбирайте верный путь с самого начала, и вы не пожалеете.

кации веб-пользователей можно применять базу данных системы FreeBSD (*/etc/passwd* и относящиеся к нему файлы), однако так поступать не следует. Ведь тогда хакерам будет намного легче проникнуть в систему, увеличится системная нагрузка и произойдут другие неприятные события, с которыми придется разбираться. Этот подход настолько плох, что я не буду рассказывать о его реализации. Но тот, кому необходимо это знать, может поискать описание в FAQ Apache.[1]

Настройка каталога

Для того чтобы иметь возможность работать с базой данных Apache, надо первым делом настроить каталог на запрос файла базы данных. Для этого создайте в каталоге файл с именем *.htaccess*, который должен выглядеть примерно так:

```
AuthName "Private Directory"
AuthType Basic
AuthDBMUserFile /usr/local/etc/apache/userdb
require valid-user
```

`AuthName` задает текст, отображаемый в поле пароля. Текст, заключенный в кавычки, можно изменить по своему усмотрению. `AuthType` сообщает серверу Apache тип аутентификации. Для стандартных имен пользователей и паролей задается тип `Basic`.

Директива `AuthDBMUserFile` сообщает серверу Apache место хранения базы данных пользователей. Не размещайте этот файл в каталоге самого веб-сайта, иначе пользователи могут загрузить его на свои машины и попробовать вскрыть. Поместите данный файл совершенно в другое место, за пределами каталога ServerRoot. Если в системе есть учетные записи, которые управляют веб-сайтами, поместите их в соответствующие домашние каталоги пользователей.

База данных веб-пользователей

Итак, расположение будущей базы определено, и настало время ее создать. Это делается с помощью утилиты dbmmanage(1). Ей требуются по крайней мере два аргумента: имя файла базы данных и команда, предписывающая действие с этим файлом.

Для создания базы данных достаточно добавить в нее пользователя; dbmmanage сообщит, что файл не существует, и создаст его. Укажите в командной строке имя пользователя, и dbmmanage выдаст приглашение на ввод пароля:

```
# dbmmanage /usr/local/etc/apache/userdbm adduser username
```

Программа dbmmanage имеет еще три аргумента – `view`, `update` и `delete`. С помощью `view` можно просмотреть список всех пользователей, присутствующих в базе, а также их зашифрованные пароли. В следущем примере пред-

[1] В любом случае вы не найдете там последовательность действий; вместо этого вам будет устроен разнос за саму мысль о таком подходе.

ставлено содержимое очень маленького файла базы данных, в которой есть только один пользователь:

```
# dbmmanage /usr/local/etc/apache/userdbm view
mwlucas:jvvAuD7bpZwY2
#
```

Для изменения пароля пользователя применяйте `update`. Удалить пользователя можно с помощью аргумента `delete`. (Для получения сведений о других аргументах обратитесь к странице документации dbmmanage(1).)

Когда база данных создана, а параметры каталога предписывают запрос паролей, у любого посетителя этого веб-каталога будут запрашиваться имя пользователя и пароль.

Apache и SSL

Многие интернет-магазины, защищенные паролями, применяют так называемые «безопасные веб-сайты». Обычно под этим понимается шифрование трафика между сервером и клиентом с помощью SSL. Хотя безопасность таких сайтов не в полной мере соответствует их названию, функции SSL жизненно важны почти для любого веб-сервера, а Apache имеет дополнительный модуль для поддержки соединений SSL. Для его применения надо установить «порт» apache13-modssl.

Каждому серверу SSL нужен сертификат. (Создание запроса на сертификат с открытым ключом рассмотрено в главе 13.) Можно создать либо настоящий сертификат, либо тестовый, но в последнем случае пользователи будут получать предупреждения при попытке просмотра страниц на веб-сервере SSL.

Полный сертификат состоит из двух частей: файла сертификата (*server.crt*) и секретного ключа (*server.key*). Разместите оба файла на веб-сервере. Для них следует предусмотреть место вне дерева каталогов ServerRoot, чтобы пользователи не могли загрузить секретный ключ. Затем с помощью chmod и chown надо разрешить чтение файла с секретным ключом только веб-серверу.

```
# chmod 600[1] server.key
# chown www server.key
#
```

Теперь сертификат появился на сервере и надо известить об этом Apache. В файле *httpd.conf*, добавленном при установке «порта» apache13-ssl, есть параметры, не отличающиеся от параметров файла настройки из стандартного «порта» Apache. В нем же есть несколько дополнительных инструкций `IfDefined SSL`, предназначенных для загрузки модулей SSL. В нижней части *httpd.conf* есть записи, определяющие файл сертификата и файл ключа:

```
SSLCertificateFile /usr/local/etc/apache/ssl.crt/server.crt
```

[1] Для того чтобы разрешить только ЧТЕНИЕ этого файла, следует выбрать права доступа 400. – *Примеч. науч. ред.*

Это путь к файлу сертификата. Он должен начинаться с косой черты (/). Помните, что любой путь в *httpd.conf*, не начинающийся с косой черты, предполагает наличие префикса DocumentRoot! Укажите здесь полный путь к файлу сертификатов (значение по умолчанию приемлемо, но при наличии виртуальных хостов его можно изменить).

Аналогично, в *httpd.conf* есть параметр, сообщающий Apache местоположение ключа сервера:

```
SSLCertificateKeyFile /usr/local/etc/apache/ssl.key/server.key
```

Файл *httpd.conf*, поставляемый в «порте» apache13-ssl, включает в себя некую «привязку», которая предписывает прослушивание соединений SSL на порту 443. Это стандартный порт TCP для защищенных соединений. Зная расположение файла сертификата и файла ключа, Apache сможет принимать соединения SSL.

Вот и все! Ваш веб-сайт с настройками по умолчанию теперь запущен и способен поддерживать безопасные сетевые соединения SSL. (Если потребуется настроить дополнительные безопасные сайты с виртуальными хостами, обратитесь к разделу «Виртуальный хостинг» далее в этой главе.)

Примечание

Лучше избегать термина «безопасный веб-сервер», если имеется в виду SSL. Хотя шифрование трафика между сервером и клиентом препятствует незаконному анализу пакетов (packet sniffing), хакеры по-прежнему могут взломать либо сервер, либо клиентскую машину. Безопасный веб-сервер подразумевает хороший дизайн веб-страниц, обслуживание сервера, а также обучение пользователей и администраторов.

Управление сервером Apache

Apache – это сложная программа, которой можно управлять по-разному. Для запуска и останова Apache (как и других программ) во FreeBSD включены сценарии, расположенные в */usr/local/etc/rc.d* (см. главу 11). В состав Apache входит своя программа управления веб-сервером.

Веб-сервером можно управлять напрямую. Например, вручную запустить сценарий запуска/останова веб-сервера (см. главу 11).

```
# /usr/local/etc/rc.d/apache.sh start
# /usr/local/etc/rc.d/apache.sh stop
```

Но такой подход имеет ограниченное применение. Для более точного управления сервером Apache применяется команда apachectl(8).

Управление сервером Apache с помощью apachectl

Запустить, остановить и проверить Apache можно с помощью специальной команды apachectl(8), предназначенной для обслуживания Apache. Хотя большинство ключей этой команды относятся к запуску и останову демона веб-сервера, она позволяет проверить конфигурацию и работу Apache. Вот эти ключи: `configtest`, `status`, `fullstatus`, `start`, `startssl`, `graceful` и `stop`.

Запустить Apache можно так:

```
# apachectl start
```

Следующая команда запускает Apache с поддержкой SSL:

```
# apachectl startssl
```

Для работы с SSL в настройках Apache должна быть предусмотрена поддержка SSL.

Следующая команда останавливает Apache, немедленно закрывая все открытые соединения:

```
# apachectl stop
```

Для выполнения «мягкого перезапуска» (graceful restart), позволяющего завершить существующие соединения HTTP, а затем перезапустить Apache, подойдет такая команда:

```
# apachectl graceful
```

Однако ее выполнение займет длительное время.

Следующая команда предоставляет экран статуса, содержащий подробную информацию о текущем состоянии сервера:

```
# apachectl fullstatus
```

Для просмотра результатов нужен lynx(1) (есть в */usr/ports/www/lynx*).

Следующая команда предоставляет отчет о состоянии сервера, созданный на скорую руку (также в lynx):

```
# apachectl status
```

А эта команда сначала вызывает configtest, а затем перезагружает Apache.

```
# apachectl restart
```

Если конфигурация приемлема, данная команда немедленно убьет все процессы httpd, а затем перезапустит Apache.

Следующая команда проверяет конфигурацию Apache и сообщает о неполадках в *httpd.conf* или в связанных с ним файлах:

```
# apachectl configtest
```

Многие другие подкоманды apachectl вызывают configtest перед выполнением тех или иных действий, чтобы убедиться в последующем успешном перезапуске Apache.

Другие способы управления демоном httpd

Программа apachectl не подходит, если запускается Apache со значительно измененной конфигурацией. Например, если комбинировать FrontPage и SSL, Apache надо запускать с аргументами, предназначенными для поддержки обоих средств. Можно применять подкоманды `configtest`, `stop`,

restart и graceful, но необходимо указать различные флаги для дополнительных модулей таким образом, чтобы их правильно интерпретировала команда apachectl. А это может оказаться делом трудоемким. В таких случаях намного проще применять команды httpd напрямую.

Хотя httpd имеет много ключей командной строки, один из них представляет для нас особый интерес. Это ключ -D, который определяет работу сервера. Например, для запуска Apache с поддержкой SSL подойдет следующая команда:

```
# httpd -DSSL
```

Для запуска Apache с поддержкой FrontPage применяйте -Dfp:

```
# httpd -Dfp
```

Легко догадаться, что эти параметры можно комбинировать:

```
# httpd -DSSL -Dfp
```

Точное описание ключей представлено в документации по соответствующим модулям Apache.

Примечание

Мне встречались серверы со значительным количеством установленных модулей. Для надлежащего запуска таких серверов требовались 9–10 аргументов -D. Если ваш сервер запускается с несколькими модулями, внесите в */usr/local/etc/rc.d/apache.sh* соответствующие исправления. Такое управление сервером предпочтительнее, чем применение apachectl.

Модули Apache

Сервер Apache состоит из модулей. Это означает, что можно добавлять или удалять фрагменты серверного кода. Apache может обслуживать такие разные расширения, как Microsoft FrontPage Extensions, языки сценариев вроде PHP и встроенный Perl. Также есть модули для сжатия страниц перед отправкой, позволяющие значительно уменьшить объем передаваемого трафика. Каждый модуль представляет собой «порт» в */usr/ports/www*. Имена файлов с «портами» модулей начинаются с «mod_», например *mod_gzip*.

Примечание

В большинстве случаев модулю необходима своя информация о настройке. Чтобы облегчить себе жизнь, размещайте конфигурационные параметры для каждого модуля в отдельных файлах и ссылайтесь на них в *httpd.conf* с помощью директив Include. Тогда при изменении модуля не составит труда найти все необходимые параметры.

Вот самые распространеннные модули Apache. Во FreeBSD каждый из них представляет собой «порт» с тем же именем (каталог */usr/ports/www*). Например, *mod_gzip* можно найти в */usr/ports/www/mod_gzip*.

mod_dav — Предоставляет сервис DAV (Distributed Authoring and Versioning).

mod_dtcl	Интегрирует в Apache интерпретатор Tcl (Tool Command Language), делая возможной быструю обработку сценариев CGI на Tcl.
mod_gzip	Сжимает данные перед отправкой, если броузер поддерживает такую возможность. Имеет смысл установить этот модуль.
mod_mp3	Превращает Apache в сервер MP3.
mod_perl	Встраивает Perl в сервер Apache, делая возможной быструю обработку сценариев CGI на Perl.
mod_php3	Предоставляет поддержку для PHP версии 3.
mod_php4	Предоставляет поддержку для PHP версии 4.
mod_python	Встраивает Python в сервер Apache для быстрой обработки сценариев CGI на Python.
mod_ruby	Встраивает Ruby в Apache, делая возможной быструю обработку сценариев CGI на Ruby.

Есть масса других модулей, но я отдаю предпочтение модулям, перечисленным выше.

FrontPage и SSL

Расширение FrontPage от Microsoft представляет особую проблему, поскольку поддержка Microsoft пакета FrontPage Extensions на платформах не-Microsoft в лучшем случае неоднородна. Если вам нужна поддержка FrontPage, установите модуль FrontPage Extensions – «порт» apache13-fp. Добавить этот модуль позднее будет очень трудно. Аналогично модуль Secure Sockets Layer (или SSL) накладывает большой отпечаток на Apache (совсем не такой плохой, как FrontPage!). Для того чтобы получить сервер с поддержкой SSL (известный еще как *https* или *Secure Web pages*), установите «порт» apache13-modssl.

Вы можете справедливо заметить, что успешное комбинирование SSL и FrontPage затруднено. Но этот путь не для неопытных или слабых духом. Не идите по нему, пока не освоите обе системы по отдельности.

Когда это время придет, возьмите новейшую версию FrontPage с *ftp://ftp.microsoft.com/products/frontpage*. Извлеките tarball и следуйте инструкциям. Столкнувшись с трудностями, проверьте архивы почтовых рассылок FreeBSD-isp. Именно там можно найти советы администраторов, уже работавших с последними версиями.

Виртуальный хостинг

Виртуальный хостинг подразумевает обслуживание одним сервером множества веб-сайтов. Сервер сконфигурирован для обслуживания веб-запросов к каждому из этих доменов, возвращая пользователям соответствующие страницы. Многим компаниям необходим очень маленький веб-сайт, содержащий лишь несколько страниц информации и, возможно, один-два сценария CGI для обработки запросов к этим данным. Именно здесь замечательно по-

дойдут виртуальные хосты. У меня были серверы FreeBSD, которые без особых усилий обслуживали тысячи маленьких доменов. Системная нагрузка не превышала 0,2. При том что каждый владелец такого сайта платит $9,95 в месяц за обслуживание пары десятков посетителей в день, можно быстро заработать хорошие деньги на недорогих аппаратных средствах.

Типичный камень преткновения в понимании виртуальных хостов – распространенное заблуждение, что волшебное слово «www» в URL указывает на веб-сайт. На самом деле, набирая URL, например *http://www.AbsoluteBSD.com*, вы предписываете веб-броузеру искать машину с именем *www.AbsoluteBSD.com*, затем подключиться к порту 80 и посмотреть, что веб-сервер сможет предложить. Набери вы «http://mail.AbsoluteBSD.com» – и клиент стал бы искать машину с этим именем.

Ухищрение, лежащее в основе виртуального хостинга, очень простое: множество имен хостов указывает на одну машину. Задача на стороне сервера состоит в том, чтобы различать запросы к различным доменам, а затем предоставлять соответствующие страницы. Возможны два различных варианта виртуальных хостов: на основе имен и на основе IP.

Виртуальные хосты на основе имен

Современные веб-броузеры, такие как Netscape 3, Internet Explorer 4 и более новые, включают в запрос имя веб-сайта, когда запрашивают веб-страницу с сервера. Благодаря этому сервер может различать запросы к сайтам, которые он обслуживает. Если известно, что клиенты наверняка не применяют Netscape 2 или Internet Explorer 3, то можно предписать серверу применять имена для идентификации виртуальных хостов. Виртуальные хосты на основе имен почти везде являются стандартом. Они должны стать стандартом и для вас (при отсутствии достаточных оснований для выбора другого варианта).

Разместите конфигурационные параметры виртуальных хостов в конце файла *httpd.conf* или даже в отдельном файле, применив ключевое слово `Include` для ссылки на него в файле *httpd.conf*. Не смешивайте конфигурацию виртуальных хостов с основной конфигурацией сервера, иначе вы запутаетесь при их рассортировке.

При конфигурировании виртуальных хостов на основе имен надо первым делом сообщить серверу Apache IP-адрес для директивы `NameVirtualHost`. Хотя виртуальные хосты на основе имен могут располагаться на одном IP-адресе, Apache должен знать, к какому IP-адресу они привязаны:

```
NameVirtualHost 192.168.33.254
```

Теперь все запросы, приходящие по этому IP-адресу, будут трактоваться как запросы к виртуальным хостам. Сейчас надо сообщить Apache, где располагается информация для каждого виртуального хоста. Как минимум, для каждого виртуального хоста следует задать имя веб-сайта и каталог, в котором располагаются документы HTML для этого сайта. Вот пример базового виртуального хоста, которому нужны только два этих элемента:

```
<VirtualHost 192.168.33.254>
  ServerName www.absolutebsd.com
  DocumentRoot /home/mwlucas/www
</VirtualHost>
```

Метки `<VirtualHost>` и `</VirtualHost>` сообщают серверу Apache, что информация между ними предназначена для одного виртуального хоста с указанным IP-адресом.

Директива `ServerName` сообщает Apache имя виртуального хоста. Сервер применяет эту запись для обработки запросов к *www.AbsoluteBSD.com*.

Наконец, запись `DocumentRoot` сообщает серверу Apache, где можно найти документы HTML, из которых состоит данный сайт.

Схема такова: когда броузер отправляет веб-запрос к виртуальному хосту по адресу 192.168.33.254 и запрашивает *www.AbsoluteBSD.com*, Apache извлекает документы из каталога */home/mwlucas/www* и отдает их клиенту.

Виртуальные хосты на основе IP

При выдаче веб-запроса не все броузеры посылают имя сайта. На самом деле в эпоху юности Интернета (Netscape 2 и Internet Explorer 3) такой порядок был общепринят. Тогда казалось, что IP-адресов так много, что они не исчерпаются. Броузеры предполагали, что им достаточно подключиться к веб-серверу на порту 80, где может располагаться только тот сайт, который им требуется.

Вы можете подумать, что такие времена прошли вместе с Apple II и Betamax, однако это не так. В 2001 году я столкнулся с корпоративной сетью, в состав которой входили 12 000 настольных компьютеров, запускающих веб-броузер на основе Mosaic в системе Windows for Workgroups. Этот броузер настолько стар, что предполагает, будто каждому веб-сайту соответствует уникальный IP-адрес. При выдаче веб-запроса он не передает имя сайта. И веб-сервер внутренней сети компании должен был поддерживать таких клиентов. Вероятно, в общедоступном Интернете такие клиенты вам не встретятся, однако необходимо уметь их конфигурировать на случай возникновения подобной ситуации.

Кроме того, сайты с поддержкой SSL подразумевают наличие единственного имени хоста для одного IP-адреса. При комбинировании SSL и виртуального хостинга неоходимо применять виртуальные хосты на основе IP-адресов.

Для применения виртуальных хостов на основе IP-адресов укажите IP-адрес в разделе `VirtualHost`, подобно варианту с виртуальными хостами на основе имен. Все выглядит точно так же, как и настройка VirtualHost для виртуальных хостов на основе имен, за исключением того, что IP-адрес в разделе `VirtualHost` — уникальный. Единственное различие в настройке виртуальных хостов двух типов состоит в том, что виртуальным хостам на основе IP не нужна директива `NameVirtualHost`.

Вот минимальная настройка виртуального хоста на базе IP:

```
<VirtualHost 209.69.178.18>
ServerName www.blackhelicopters.org
DocumentRoot /home/mwlucas/www2
</VirtualHost>
```

Любой запрос, поступающий на порт 80 по IP-адресу 209.69.178.18, получит документы HTML из */home/mwlucas/www2*.

Тонкая настройка виртуальных хостов

Обеспечив работу виртуальных хостов с минимальной конфигурацией (описанных в предыдущих разделах), можно провести дополнительную настройку. Здесь мы обсудим различные параметры виртуальных хостов, которые подходят для виртуальных хостов обоих типов.

Номера портов

Различные документы можно обслуживать на разных портах. (Вероятно, вы сталкивались с этим раньше, когда имя хоста в URL завершалось двоеточием и числом.) Например, если Apache прослушивает порты 80 и 81, на каждом из них можно разместить различные виртуальные хосты, если добавить номера портов к IP-адресам в директиве `VirtualHost`.

Например, вот конфигурация для создания двух различных сайтов (*http://www.AbsoluteBSD.com* и *http://data.AbsoluteBSD.com:8080*), предусматривающая два похожих виртуальных хоста на различных портах. Оба сайта располагаются на одном IP-адресе, но на разных портах:

```
<VirtualHost 209.69.178.18:80>
ServerName www.AbsoluteBSD.com
DocumentRoot /home/mwlucas/www
</VirtualHost>
<VirtualHost 209.69.178.18:8080>
ServerName data.AbsoluteBSD.com
DocumentRoot /home/mwlucas/data
</VirtualHost>
```

Протоколирование виртуальных хостов

По умолчанию работа виртуальных хостов находит отражение в файле протокола сервера Apache, однако для виртуальных хостов можно предусмотреть отдельные файлы. (Это обычная практика для коммерческих веб-серверов, где отдельный файл протокола может потребоваться каждому пользователю.) Здесь на помощь приходят директивы `ErrorLog` и `TransferLog`, каждая из которых воспринимает имя файла протокола как аргумент:

```
<VirtualHost 209.69.178.18:80>
ServerName www.AbsoluteBSD.com
DocumentRoot /home/mwlucas/www
ErrorLog /home/mwlucas/absolutebsd.com-error-log
TransferLog /home/mwlucas/absolutebsd.com-access-log
</VirtualHost>
```

Директива TransferLog в качестве второго аргумента может принимать тип информации, подлежащей протоколированию. Стандартные стили ведения протокола рассматривались при обсуждении директивы LogFormat (в разделе «Формат протокола») ранее в этой главе.

Options и AllowOverride

По умолчанию виртуальные хосты наследуют значения Options и AllowOverride из корневого каталога. Ниже приведен пример конфигурации корневого каталога (просто для освежения в памяти):

```
<Directory />
    Options AuthConfig Limit
    AllowOverride none
</Directory>
```

Для всех веб-сайтов и всех каталогов на этом сервере по умолчанию установлены параметры AuthConfig и Limit и ничего не разрешено в AllowOverride. Чтобы подменить эти значения для виртуального хоста, задайте Options и AllowOverride в соответствующем разделе виртуального хоста. (Любой параметр, пригодный для конфигурации основного хоста, можно применять для виртуальных хостов.)

С помощью подмены параметров можно варьировать доступ к серверу в зависимости от клиента. Например, разрешить SSI на одном виртуальном хосте и запретить на остальных. Значения Options следующего виртуального хоста подменят значения сервера по умолчанию:

```
<VirtualHost 209.69.178.18>
ServerName data.AbsoluteBSD.com
DocumentRoot /home/mwlucas/data
Options Limit IncludesNOEXEC
</VirtualHost>
```

Итак, вы научились конфигурировать виртуальные хосты для клиентов, но им еще может понадобиться загружать файлы на сервер (обычно через FTP).

.NET во FreeBSD

.NET – это набирающая популярность технология написания программ для Сети. (Формально официальной платформой .NET компании Microsoft является .NET Server.)

.NET представляет собой реализацию ECMA Common Language Infrastructure (CLI) – стандарта, созданного и поддерживаемого Microsoft. Компания Microsoft выпустила отдельную реализацию CLI, работающую как во FreeBSD, так и в Windows 2000. Продукт называется Shared Source Common Language Infrastructure (SSCLI), а его кодовое название – *Rotor*. Предположительно, версия .NET Server интегрирована с другими продуктами Microsoft более тесно, нежели версия Rotor, распространяемая с ограничениями (shared-source).

Когда эта книга готовилась к печати, пакет Rotor был доступен в виде бета-версии. Видимо, он будет регулярно обновляться.[1] В данном разделе обсуждаются основы установки SSCLI в системе FreeBSD.

Установка SSCLI

Пакет SSCLI доступен в */usr/ports/lang/cli*. В настоящее время «порт» будет собран должным образом лишь при условии применения командной оболочки /bin/sh. Если вы применяете другую командную оболочку, перейдите на /bin/sh для установки этого «порта». Затем запустите такие команды:

```
# cd /usr/ports/lang/cli
# exec /bin/sh
# export SHELL=/bin/sh
# make install
```

Програмы будут загружены и скомпилированы, но не установятся, поскольку Microsoft разрешает использовать это программное обеспечение только в рамках лицензии. Когда компиляция завершится, будет выдано извещение о необходимости прочесть лицензионное соглашение, которое находится в */usr/ports/lang/cli/work/sscli/license.txt*. Прочтите это соглашение и уясните его смысл. В обычаях Microsoft лицензировать программное обеспечение таким образом, что условия соглашения могут стеснять специалистов, привыкших к FreeBSD. Кроме того, Microsoft изменяет соглашения так, как считает нужным.

Прочитав лицензию, запустите такую команду:

```
# make -DI_AGREE_TO_LICENSE_TERMS install
```

Примечание

Я рекомендую не выполнять `make clean` при установке. В пакет Rotor входит много исходного кода, документации и примеров, которые могут понадобиться при работе с этим экспериментальным программным обеспечением. Команда `make clean` удалит большую часть перечисленного.

Предыдущая команда установит SSCLI в */usr/local* – в подкаталоге, название которого зависит от устанавливаемой версии Rotor. (Первая бета-версия устанавливалась в */usr/local/cli-20020326*.) После того как вы добавите этот каталог в `$PATH`, все будет готово для запуска Rotor.

FTP

FTP – это классический протокол для переноса файлов с одного компьютера на другой по Интернету. Большинство пользователей выберут его для пере-

[1] SSCLI стал версией 1.0 (официальный выпуск) 05.11.2002. Подробности и файлы для скачивания есть по адресу *http://www.microsoft.com/downloads/details.aspx?FamilyId=3A1C93FA-7462-47D0-8E56-8DD34C6292F0&displaylang=en*. – *Примеч. науч. ред.*

дачи файлов на сервер и загрузки файлов с сервера. Как и другие почтенные протоколы, FTP от времени не стал лучше. Его проблемы многочисленны, а их решения встраивались в протокол годами. Хотя система FreeBSD обеспечивает максимально легкое обслуживание FTP, необходимо предпринять определенные шаги, чтобы этот протокол работал надлежащим образом.

Защита FTP

Пароли и имена пользователей передаются по FTP открытым текстом, поэтому обладатель «сниффера» пакетов может перехватить имена и пароли FTP. Никто в локальной сети, за исключением сетевого администратора, не должен располагать такой возможностью. В этом случае с постоянными пользователями все в порядке. Однако пользователи могут настаивать на FTP-доступе.

Не передавайте по сети пароль системного администратора открытым текстом, а загружайте и выгружайте файлы с помощью scp(1). (scp рассматривается в конце главы.) Программа scp не слишком популярна (пока), а при наличи большого количества открытых соединений шифрование, применяемое scp, может привести к перегруженности сервера.

Клиент FTP

FTP – достаточно сложный протокол. В отличие от POP3 и SMTP, его трудно тестировать. Для работы с FTP нужен FTP-клиент. В составе FreeBSD такой клиент есть. Для подключения к хосту достаточно набрать `ftp` и имя хоста.

```
# ftp magpire.blackhelicopters.org
Connected to magpire.blackhelicopters.org.
220 magpire.blackhelicopters.org FTP server (Version 6.00LS) ready.
Name (magpire:mwlucas):
```

По умолчанию клиент посылает локальное имя пользователя, однако при необходимости можно ввести другое имя. Далее будет запрошен пароль:

```
331 Password required for mwlucas.
Password:
230 User mwlucas logged in, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Если все идет, как задумывалось, то в результате вы зарегистрируетесь на удаленном сервере и войдете в оболочку, подобную командной оболочке. Выполнять команды вы не сможете, но сможете перемещаться и просматривать файлы с помощью стандартных команд UNIX, таких как ls и cd.

Загрузка по FTP

Для копирования файла с удаленного сервера на ваш локальный сервер применяйте команду get:

```
ftp> get .cshrc
local: .cshrc remote: .cshrc
150 Opening BINARY mode data connection for '.cshrc' (767 bytes).
100% |***************************************************|   767       00:00 ETA
226 Transfer complete.
767 bytes received in 0.00 seconds (173.42 KB/s)
ftp>
```

Как видите, клиент FTP открывает соединение для загрузки файла. По мере скачивания файла по экрану движется строка из звездочек (*). Значение в поле ETA обновляется в соответствии с временем, оставшимся до окончания загрузки. По завершении передачи файла вы увидите извещение, размер загруженного файла и приглашение на ввод команды FTP.

Загрузка файлов на удаленный компьютер через FTP

Для копирования файла с локальной системы на удаленную применяйте команду put. Ее вывод очень похож на вышеприведенный пример, поэтому я не буду его повторять.

Передача нескольких файлов

Для одновременной передачи нескольких файлов применяйте mget и mput. Например, чтобы передать все файлы с расширением *.txt* из текущего каталога, надо выполнить такую команду:

```
ftp> mget *.txt
```

Блокирование приглашений

По умолчанию перед копированием каждого файла FTP предлагает подтвердить ваше намерение и отображает соответствующее приглашение. Но кому захочется подверждать передачу каждого файла, если в каталоге их десятки или даже сотни? Включать и выключать верификацию можно с помощью команды prompt:

```
ftp> prompt
Interactive mode off.
ftp>  mget *
```

В данном примере будут загружены все файлы текущего каталога, причем запросы на верификацию выдаваться не будут.

Передачи двоичных и ASCII-файлов

Различие между передачей двоичных и ASCII-файлов – постоянный источник путаницы из-за того, что по-разному обрабатываются символ возврата и символ разделителя строк. С давних пор в системах DOS и UNIX конец строки обозначался по-разному, что можно увидеть при передаче файлов между двумя системами. В Интернете можно найти много документов, детально описывающих это различие. Есть много статей, в которых сторонники двух систем жестко критикуют противоположные подходы. Вам же надо лишь знать, как действовать в сложившейся обстановке.

Системы UNIX по умолчанию подразумевают двоичную передачу, а системы Windows – передачу ASCII. Команда `bin` предписывает серверу FTP выполнять двоичную передачу, а команда `a` – передачу ASCII.

На самом деле в двоичном режиме можно с успехом передавать как двоичные, так и ASCII-файлы. В режиме ASCII можно передавать файлы ASCII, однако двоичные файлы, передаваемые в режиме ASCII, будут повреждены. Двоичный режим подходит для обоих случаев, поэтому применяйте именно его.

Просмотр файлов

Наконец, поговорим о том, как просматривать файлы. Команда `less` постранично отображает содержимое файла, позволяя сканировать индексные и readme-файлы для принятия обдуманного решения о загрузке тех или иных файлов.

Сервер FTP

Как передавать файлы по FTP, мы уже знаем и теперь рассмотрим вопросы предоставления сервиса FTP.

По умолчанию FreeBSD запускает демон FTP из inetd. Большинство систем не получают много запросов FTP, поэтому inetd легко справляется с поступающими запросами. Если не требуется поддерживать большое количество одновременных сессий FTP, то подойдут значения по умолчанию.

Однако если вы запускаете высокомощный сервер FTP и собираетесь обслуживать десятки, сотни и даже тысячи одновременных соединений, то вызов FTP из inetd приведет к дополнительной нагрузке на систему. Работа системы замедлится из-за inetd, и пользователи будут недовольны.

Для преодоления этих трудностей надо убрать ftpd из inetd. Настройте ftpd для запуска в *автономном* (*standalone*) режиме, который подразумевает постоянное прослушивание сети и самостоятельное обслуживание запросов.

Настройка автономного ftpd

Для запуска ftpd в автономном режиме первым делом отключите вызов ftpd из inetd, закомментировав соответствующую строку в */etc/inetd.conf*. Мы поступаем так, поскольку лишь одна программа может задействовать порт TCP в тот или иной момент. Поскольку inetd запускается раньше любых локальных программ, он заранее сообщит системе, что прослушивает все входящие соединения с TCP-портом FTP. В этом случае запустить автономный ftpd не удастся.

Для запуска ftpd в автономном режиме предназначен ключ `-D`. Подобно другим автономным и дополнительным программам, запускать ftpd автоматически можно с помощью сценария */usr/local/etc/rc.d*:

```
#!/bin/sh

/usr/libexec/ftpd -D
```

Теперь пользователи могут устанавливать соединения FTP и передавать файлы. Именно так! Все остальные действия представляют собой тонкую настройку сервера FTP.

Протоколирование обращений к ftpd

Демон ftpd имеет различные функции протоколирования. Если задать ключ `-l` один раз (значение по умолчанию в */etc/inetd.conf*), ftpd будет протоколировать все успешные и неуспешные попытки регистрации. Если данный ключ указать дважды, ftpd будет протоколировать все операции FTP: загрузку файлов на сервер и с сервера, создание и удаление каталогов и любые операции с файлами.

Добавьте эти ключи в конец командной строки ftpd, представленной в файле запуска. Строка, позволяющая запустить ftpd в автономном режиме и протоколировать все операции FTP, будет выглядеть так:

```
/usr/libexec/ftpd -D -ll
```

Запрет изменений на сервере

Иногда требуется запретить загрузку файлов на сервер или любые другие операции, приводящие к изменениям в файловой системе сервера. На корпоративном сервере документации и сайте-зеркале лучше предоставлять только услугу загрузки файлов с сервера. Для этого применяется ключ `-r`.

Клетки для пользователей

Типичная проблема FTP заключается в том, что если пользователям разрешено загружать с сервера любые файлы, то пароли FTP могут быть похищены. Разумно ли подвергать свою систему риску из-за того, что какой-то пользователь-разиня не уберег свой пароль!

Чтобы запретить пользователю рыскать по системе, можно посадить его в *«клетку»* (*cage*) или применить *chroot*. Клетка представляет собой домашний каталог пользователя. Пользователь, помещенный в клетку, может загружать, выгружать или изменять файлы только в своем домашнем каталоге (или его подкаталогах). Если пользователь пытается войти в каталог вне клетки, система не разрешит такой доступ. Это очень похоже на маленькую клетку, рассмотренную в главе 8.

Вариант с chroot полезен для веб-серверов с большим количеством клиентов на одной машине. В конце концов, только пользователь должен видеть свой каталог и больше никто.

chroot для пользователей

Для того чтобы посадить пользователя в клетку, добавьте его имя в файл */etc/ftpchroot*. Всякий раз, когда пользователь подключается к системе по FTP, его учетная запись сравнивается с содержимым */etc/ftpchroot*. Если имя есть в этом файле, пользователь блокируется в своем домашнем каталоге и получает возможность полного контроля над ним, – создавать столько

подкаталогов и хранить столько файлов, сколько позволяет отведенное ему дисковое пространство. Но пользователь не может покидать свой домашний каталог и исследовать систему.

chroot для групп пользователей

Пользователи, входящие в группу, указанную в */etc/ftpchroot*, также оказываются в клетке. (Перед именем группы ставится знак (@).) Вы можете указать группу в */etc/login.access* и запретить регистрацию по SSH. Тогда все пользователи этой группы будут закованы достаточно крепко.

Допустим, что в системе есть два младших системных администратора, Фил и Крис. Требуется разрешить им загрузку файлов только в свои домашние каталоги. Есть также группа клиентов, которые поддерживают свои веб-сайты. Клиенты входят в группу webclients, указанную в */etc/group*. Чтобы «посадить» всех этих пользователей в клетку, настройте */etc/ftpchroot* так:

```
phil
chris
@webclients
```

Запрет FTP-доступа

Название файла */etc/ftpusers* обманчиво. Файл содержит не список разрешенных пользователей, а список тех, кому *не* позволено входить в систему по FTP. По умолчанию в */etc/ftpusers* представлены различные учетные записи, такие как root и nobody.

В этот файл можно добавить группы, поместив перед их именами символы @. Например, FTP-доступ можно запретить членам группы wheel. (Пользователи, применяющие пароль root, не должны передавать свои пароли открытым текстом!)

Сообщения при подключении

Когда клиент устанавливает соединение FTP, на его экране отображается содержимое файла */etc/ftpwelcome*. В него можно поместить предупреждения, сообщения о предоставляемых возможностях, непристойности, угрозы – что угодно. Пользователи увидят данную информацию еще до получения приглашения к регистрации. Это замечательное место для сообщения о «запрете несанкционированного использования». Факт наличия сообщения пригодится в суде, если кто-нибудь воспользуется вашими услугами в противозаконных целях.

Приветственные сообщения

Когда пользователь входит в систему, отображается содержимое */etc/ftpmotd*. Обычно в нем содержатся соглашения по использованию.

Настройка анонимного сервера FTP

Анонимные FTP-сайты – это популярный способ предоставления файлов и документов для всего Интернета. Однако такие сайты зачастую взламыва-

ются. Хотя ftpd системы FreeBSD достаточно устойчив и безопасен, необходимо предпринять меры предосторожности и настроить сервер должным образом. Вот несколько рекомендаций:

1. Если возможно, разрешите только считывание файлов с сервера, применив ключ `-r`.
2. Применяйте ключ `-S` для протоколирования в */var/log/ftpd* всех операций анонимного сервера FTP. Файл следует создать до начала протоколирования, поэтому надо выполнить `touch /var/log/ftpd`.
3. Создайте пользователя с именем «ftp». Домашним каталогом этого пользователя будет корень дерева каталогов анонимного сервера FTP. Все файлы надо разместить в этом каталоге.
4. Создайте каталог */home/ftp/pub* для папки *pub*, традиционно присутствующей на серверах FTP. Если необходимо разрешить пользователям загружать файлы на сервер, установите для этого каталога режим 777 (`chmod 777 pub`). Для того чтобы разрешить только считывание, установите режим 444. Такие настройки обеспечат работу базового анонимного сервера FTP.

Предостережение по поводу анонимного FTP

Разрешение всем желающим загружать файлы на сервер кажется дружеским шагом. У вас есть достаточная полоса пропускания и вы хотите предоставлять общедоступный сервис. Это замечательно, но ведь мир не идеален.

Однако если разрешить всем желающим хранить данные на вашей системе, на ней могут разместиться нелегальные программы, детская порнография или переписка террористов. Чтобы все это было трудно обнаружить, злоумышленники могут создать скрытые каталоги или замаскировать данные. Даже если вы отслеживаете все барахло, загружаемое пользователями, вы можете не обратить внимание на файл *FTPd.configuration.txt*; вы не сможете проверить каждый файл. Это не ваша вина, что в этом файле скрыт учебный MPEG-фильм Фронта Освобождения Хомяков, имеющих секретное задание засадить президента в колесо и заставить его бегать в нем всю оставшуюся жизнь.

Примечание

В состав FreeBSD входят различные серверы FTP, которые находятся в */usr/ports/ftp*. Многие из них имеют отличные функции, интересные сайты и сообщества преданных пользователей. Обратитесь к архивам почтовой рассылки freebsd-security и почитайте мнения о том или ином сервере, перед тем как установить его. Дело в том, что в системах безопасности многих популярных FTP-серверов зияли бреши. Например, один полулярный сервер FTP имеет неофициальный подзаголовок «Предоставление удаленного доступа с правами root начиная с 1994 года». Оглянитесь вокруг и будьте разборчивы.

16

Файловые системы и диски

Важность управления файловыми системами и дисками трудно переоценить. Для операционной системы надежность и гибкость работы с дисками первостепенны, потому что на дисках хранятся ваши данные. FreeBSD предоставляет выбор файловых систем и различных вариантов их обслуживания. В этой главе мы рассмотрим типичные компоненты, с которыми имеет дело системный администратор.

Файлы устройств

Файлы устройств – это специальные файлы, представляющие аппаратные средства системы. Они служат логическим интерфейсом между пользовательскими программами с одной стороны и драйверами устройств или физическими устройствами с другой. Выполняя команду с файлом устройств, посылая информацию файлу устройств или читая данные из него, операционная система выполняет те или иные операции с физическим устройством. Для разных устройств эти операции могут существенно различаться, – в конце концов, запись данных на диск дает совсем другие результаты, нежели передача данных звуковой карте! Файлы устройств можно найти в каталоге */dev*.

Для того чтобы приступить к работе с диском или дисковыми разделами, необходимо знать имя устройства или имя файла, который представляет это устройство. Зачастую имена файлов устройств непонятны. Как правило, они основаны на имени драйвера соответствующего аппаратного средства, а имена драйверов устройств, в свою очередь, основаны на названии микросхемы, применяемой в устройстве, а не на его назначении. В */dev* можно найти много различных файлов устройств.

Большинство файлов устройств описаны на соответствующих страницах руководства. Наибольший интерес для нас представляют файлы дисковых устройств. В следующем списке представлены типичные файлы устройств, с которыми вы столкнетесь. (Для получения дополнительной информации о каждом из этих файлов обратитесь к страницам руководства.)

Файл устройства	Описание
/dev/fd*	Дискеты
/dev/acd*	CD-ROM с интерфейсом IDE
/dev/ad*	Жесткие диски IDE и разделы
/dev/cd*	CD-ROM с интерфейсом SCSI
/dev/da*	Жесткие диски SCSI и разделы[a]

[a] Во FreeBSD до 2.2.x включительно файлы устройств дисков SCSI назывались */dev/sd**, и автор далее использует это название. Сейчас в версиях 4.x и 5.x употребляется обозначение */dev/da**, хотя если производилось обновление системы с версии 2.2.x, диски все равно будут называться *sd**. – *Примеч. науч. ред.*

Жесткие диски и разделы

Прежде всего, рассмотрим диски IDE. Наш первый диск IDE называется */dev/ad0*. За ним следуют диски */dev/ad1*, */dev/ad2* и т. д. Подразделы на каждом диске начинаются с этого имени.[1] К данному названию добавляются дополнительные символы, например */dev/ad0s1b*. Диск можно считать одним целым, но в каталоге */dev*, как правило, есть масса файлов подразделов, начинающихся с */dev/ad0*:

```
#ls /dev/ad*
/dev/ad0        /dev/ad0s1b     /dev/ad0s1f     /dev/ad0s1h
/dev/ad0s1a     /dev/ad0s1e     /dev/ad0s1g
#
```

Итак, что означают все эти подразделы? Вспомните, как вы распределяли дисковое пространство для FreeBSD. Если вы следовали рекомендациям этой книги, весь диск отдан FreeBSD. А может быть, вы создали второй раздел для другой операционной системы или даже разбили диск на две секции FreeBSD. В мире Microsoft и Linux эти секции называются *разделами* (*partitions*), а на территории FreeBSD – *участками* (*slices*). Символы s1 в предыдущем листинге /dev/ad* представляют большие разделы, или участки. Диск, представленный в листинге, имеет один участок, который разбит на подразделы, отмеченные символами.

Во FreeBSD *раздел* – это сегмент внутри участка. Разделы внутри участка создаются во время установки системы. Каждый раздел имеет уникальный файл устройства, название которого образуется добавлением уникального символа к названию файла участка. Например, разделы внутри участка */dev/ad0s1* представлены как */dev/ad0s1a*, */dev/ad0s1b*, */dev/ad0s1e* и т. д. Каждому разделу, создаваемому во время установки системы (*/usr*, */var* и т. д.), назначается соответствующий файл устройства.

[1] В компьютерах обычно первый диск имеет номер устройства 0, второй диск – номер устройства 1 и т. д.

Название файлу устройства, соответствующего тому или иному разделу, можно назначать произвольным образом, однако есть исключения. Традиционно файл устройства, имя которого заканчивается на «a» (в нашем примере */dev/ad0s1a*), есть корневой раздел (root), а файл с замыкающим символом «b» (*/dev/ad0s1b*) предназначен для раздела свопа. Метка «c» служит признаком всего участка, от начала до конца, а символы от «d» до «h» можно назначить любому разделу по своему усмотрению. В одном участке может быть не больше восьми разделов, а на одном диске – не больше четырех участков.

Например, файл устройства */dev/ad0s1a* соответствует диску с номером 0, участку 1, разделу 1 и, возможно, файловой системе root. Файл устройства */dev/ad1s2b* указывает на диск 2 и, возможно, участок свопа.

Участки и разделы на дисках SCSI подобны участкам и разделам на дисках IDE, однако их файлы устройств немного различаются. Имена файлов устройств, соответствующих жестким дискам SCSI, начинаются с */dev/sd*, а не с */dev/ad*. Например, второму участку третьего жесткого диска SCSI соответствует файл */dev/sd2s2*.

Примечание

Время от времени вам будет встречаться слово «традиционный». Это означает обычный порядок вещей. По возможности следует придерживаться традиционных решений. Ведь вы не хотите, чтобы из-за нетрадиционной компоновки диска системный администратор разместил раздел свопинга на разделе с вашими данными?

Файл /etc/fstab

В таблице файловых систем перечислена каждая файловая система, расположенная на жестких дисках компьютера. Показаны точки монтирования каждой файловой системы и специальные параметры, которые применяет mount(8). В файле */etc/fstab* каждая файловая система представлена в отдельной строке. Вот пример:

```
#more /etc/fstab
# Device              Mountpoint     FStype  Options        Dump   Pass#
/dev/ad0s1b           none           swap    sw             0      0
/dev/ad0s1a           /              ufs     rw             1      1
/dev/ad0s1f           /test1         ufs     rw             2      2
/dev/ad0s1g           /test2         ufs     rw             2      2
/dev/ad0s1h           /usr           ufs     rw             2      2
/dev/ad0s1e           /var           ufs     rw             2      2
/dev/acd0c            /cdrw          cd9660  ro,noauto      0      0
/dev/acd1c            /cdrom         cd9660  ro,noauto      0      0
proc                  /proc          procfs  rw             0      0
#
```

Первое поле в этом листинге сообщает имя устройства, уже обсуждавшееся ранее.

Второе поле указывает точку монтирования (каталог, в котором можно найти файловую систему). Обычно это */usr*, */var* и т. д., хотя специальные разделы, например разделы свопинга, могут не иметь точки монтирования (`none`).

Далее следует тип файловой системы. Для стандартных разделов FreeBSD применяются файловые системы типа `FFS` (Fast File System, быстрая файловая система, ufs). Другие варианты: `msdos` (разделы FAT), `mfs` (Memory File System) и `cd9660` (CD-ROM). Этими файловыми системами список не исчерпывается. Перед монтированием файловой системы надо выяснить, как она форматирована. (Легко догадаться, что попытка монтировать дискету DOS как файловую систему UNIX не даст удовлетворительных результатов.)

В четвертом поле представлены параметры mount, которые применяются для данной файловой системы. Они описывают, как ядро должно интерпретировать файловую систему. Далее параметры монтирования будут обсуждаться более подробно, а здесь рассмотрим несколько специальных параметров, применяемых только в */etc/fstab*:

`ro`

Файловая система монтируется только для чтения. Даже root не может осуществлять запись в файлы этой файловой системы.

`rw`

Файловая система монтируется для чтения-записи. Это стандартное монтирование `noasync` (см. раздел «Типы монтирования FFS»).

`noauto`

Раздел не будет автоматически монтироваться во время начльной загрузки или при запуске `mount -a`. Этот параметр применяется для приводов CD-ROM накопителей на гибких дисках, в которых может не оказаться носителя.

`Dump`

Сообщает программе dump(8), надо ли в этой файловой системе выполнять дамп. Если значение в этом поле равно 0, программа dump не будет сохранять файловую систему. Если в поле стоит ненулевое значение, то заданное число определяет минимальный уровень дампа, необходимый для сохранения файловой системы. (Для получения более подробной информации обратитесь к главе 3.)

`Pass#`

Сообщает системе, где следует монтировать файловую систему во время начальной загрузки. Файловые системы с `Pass#`, равным 0, не будут автоматически монтироваться при загрузке. Значение 1 имеет только корневая файловая система. Она монтируется первой. Для всех других файловых систем значение поля `Pass#` равно 2. Это означает, что они должны монтироваться после монтирования корневой файловой системы.

В предыдущем примере */dev/ad0s1b* есть раздел свопа, а */dev/ad0s1e* монтируется как */var*. Можно заметить наличие двух приводов CD-ROM: */dev/acd0c* и */dev/acd1c* (один из них записывающий). Для управления дисковыми разделами необходимо знать соответствие разделов и файлов устройств.

Основы организации диска

Все файловые системы имеют что-то общее. Не последнее место в списке занимает сам дисковый накопитель – прямоугольное устройство с разъемами на одном конце.

Дисковый накопитель – это маленький волшебный блок. Когда с ним обращаются плохо, он издает магический визг и скрежет. Если с ним обращаться еще хуже, из него может пойти волшебный дым, и больше диск работать не будет. Для настоящего понимания файловых систем необходимо получить представление о колдовских чарах, скрывающихся внутри волшебного блока. (Если у вас есть старый пыльный диск, который вам больше не нужен, разберите его и посмотрите, как он устроен.)

Примечание

Ниже приведено лишь общее описание жесткого диска. Вероятно, многие из вас знают о них намного больше.

Сняв корпус винчестера, можно увидеть набор круглых дисков, обычно называемых *пластинами* (*platters*). Когда диск включен, эти пластины (обычно изготовленные из стекла или пластика) вращаются. Скорость вращения пластин составляет тысячи оборотов в минуту (Revolutions Per Minute, RPM).

Пластины покрыты слоем магнетика, который, в свою очередь, обычно покрыт оксидом железа.[1] Слой магнетика образует тысячи круговых колец, называемых *дорожками* (*tracks*), которые заполняют пластины от середины к внешнему краю, как годичные кольца в дереве. Дорожки хранят данные в виде строк из нулей и единиц. Каждая дорожка разбита на *сектора* (*sectors*). Каждый сектор на внешних дорожках хранит больше данных, чем такой же сектор на внутренней дорожке. Кроме того, чтение данных на внешней дорожке занимает меньше времени, чем чтение того же объема данных на внутренней дорожке, так как любая точка на внешней дорожке перемещается быстрее.

Головки располагаются над каждой пластиной. Через них проходят все данные при чтении информации с пластин и записи информации на пластины. Как правило, головки могут быстро читать и записывать данные, однако они должны ждать, пока пластины не пройдут под ними, чтобы можно было передать данные. По сути, производительность накопителя определяется скоростью, с которой пластины перемещаются под дисковыми головками. Вот почему значение RPM так важно.

[1] Оксид железа – это ржавчина. Пожалуйста, не пытайтесь увеличить емкость пластин, посыпая их ржавчиной. Это не помогает, что я доказал много лет тому назад.

Каждая дорожка хранит блоки данных. Их размер и местоположение определяют эффективность файловой системы и виды файлов, которые файловая система может обслуживать наилучшим образом. Каждая файловая система применяет свою схему индексирования для записи информации о размещении данных на пластинах. Одна файловая система вовсе не обязательно сможет прочитать индекс другой. (В системе DOS применялась только File Allocation Table (FAT). Впоследствии эта таблица расширилась в FAT12, FAT16, FAT32 и т. д.) В свою очередь, в UNIX в файловой системе Fast File System (FFS) применяются *индексные дескрипторы* (*index nodes*, или *inodes*), распределенные по всему диску.

Fast File System

Файловая система Fast File System (FFS) – прямая наследница файловой системы, поставляемой с BSD4.4. (Фактически на момент написания этих строк один из авторов исходной файловой системы продолжает дорабатывать файловую систему FreeBSD, добавляя отличные функции, которые будут вкратце рассмотрены.) Иногда FFS называют UFS (UNIX File System). Многие системные утилиты по-прежнему называют разделы FFS разделами UFS.

Примечание

Из BSD файловая система FFS перешла к другим производителям. Если производитель UNIX не расхваливает свою «улучшенную, усовершенствованную» файловую систему, то почти наверняка это файловая система BSD.

При разработке FFS предполагалось создать быструю и надежную файловую систему, которой по силам эффективно справляться с обычными (и необычными) ситуациями. FreeBSD поставляется с FFS, которая сконфигурирована в расчете на извлечение максимальной пользы из современных аппаратных средств, однако FFS можно оптимизировать для обслуживания триллионов маленьких файлов или полудюжины файлов размером 30 Гбайт. Знать детали внутренней организации FFS необязательно, но основы понимать надо. Прежде всего, полезно знать о том, что FFS делит диск на индексные дескрипторы (inodes) и блоки (blocks).

Индексные дескрипторы

Содержат базовую информацю о файлах, включая права доступа, размер и т. д., а также список блоков в этом файле. Данные, хранящиеся в индексных дескрипторах, называются метаданными. Это просто данные о данных (*metadata*).

Блоки

Содержат фактические данные файла.

В индексных дескрипторах хранится много различной информации, но вам достаточно знать, что они представляют собой *индексы* файлов на диске. С их помощью операционная система находит информацию, которую пытается извлечь.

Виртуальные индексные дескрипторы

Во времена молодости UNIX индексные дескрипторы и блоки работали замечательно. Однако с годами вошло в практику перемещать диски между различными машинами и даже различными операционными системами. Стали популярны приводы CD-ROM, у которых носители имеют свою собственную уникальную компоновку. Для дискет файловая система FAT32 стала стандартом. В других UNIX разработаны свои варианты файловых систем. Поскольку BSD надо было «общаться» со всеми этими и другими системами, потребовался иной уровень абстракции.

Такая абстракция была реализована в виде виртуальных индексных дескрипторов (*virtual nodes*), или *vnodes*. Виртуальными дескрипторами никогда не манипулируют напрямую, однако ссылки на них есть в документации системы. Следовательно, важно знать, что они собой представляют. Виртуальный дескриптор играет роль переводчика информации, передаваемой между ядром и той или иной файловой системой. Если файл записывается в файловой системе FFS, виртуальный дескриптор обращается к индексному, а если файл записывается в системе FAT (Microsoft), виртуальный дескриптор обращается к таблице размещения файлов. (На самом деле у виртуальных дескрипторов гораздо больше функций, однако здесь они не будут рассматриваться.)

Каждый инструмент, который обращается к диску для чтения или записи данных, действует через виртуальные дескрипторы, которые устанавливают соответствие между данными и файловой системой на физическом носителе. Ссылки на индексные дескрипторы (inodes) можно встретить только в связи с файловыми системами FFS, а виртуальные дескрипторы (vnodes) применяются в работе с любой файловой системой.

Типы монтирования FFS

В отличие от файловых систем Windows или Macintosh разделы FFS можно интерпретировать по-разному в зависимости от варианта монтирования. Способ монтирования раздела называется *типом монтирования* (*mount type*). Как было сказано в разделе «Файлы устройств» ранее в этой главе, для монтирования файловой системы надо знать имя физического устройства. К немонтированной системе нет доступа – файлы нельзя читать, записывать или просматривать каким бы то ни было способом. Это тупик.

Для изменения параметров монтирования раздела укажите их в соответствующей строке файла */etc/fstab* в столбце `options`.

Монтирование «только для чтения»

Если надо только просматривать содержимое диска и нет необходимости записывать данные, можно монтировать раздел «только для чтения» (read-only, или `rdonly`). Бесспорно, это самый безопасный способ монтирования диска. В то же время это один из бессмысленных способов для большинства серверных приложений, поскольку невозможно изменять данные или записывать новые.

Многие системные администраторы монтируют «только для чтения» корневой раздел (root), и возможно, даже */usr*, чтобы минимизировать ущерб, наносимый выключением электропитания. Даже если вы «потеряете» физический жесткий диск при всплеске напряжения либо из-за других неполадок в оборудовании, данные на пластинах останутся невредимы. В этом заключается преимущество монтирования «только для чтения»; недостатком является сложность эксплуатации такой файловой системы, поскольку невозможно записывать данные на диски, монтированные «только для чтения»!

Синхронное монтирование

Синхронное (sync) монтирование – это старинный способ монтирования файловой системы. При этом можно свободно читать данные с диска, однако когда данные записываются, ядро ждет завершения данной операции, прежде чем сообщить программе о выполнении записи. Если запись не завершена, программа вправе предпринимать соответствующие действия.

Синхронное монтирование обеспечивает целостность данных в случае аварии, однако работа с такой файловой системой происходит медленно. (Понятие «медленно» в наши дни относительно; даже дешевый диск может превосходить диски, которые считались лучшими несколько лет назад.) Если вы скрупулезно относитесь к соблюдению целостности данных, задумайтесь о применении синхронного монтирования. Однако в большинстве случаев это избыточная мера.

Асинхронное монтирование

Для ускорения доступа к данным (с более высоким риском их потери) монтируйте разделы асинхронно (async). Когда диск монтирован асинхронно, ядро записывает данные на диск и сообщает программе о том, что запись прошла успешно. При этом ядро не ждет от диска подтверждения того, что данные действительно записаны. Асинхронное монтирование – прекрасный выбор для второстепенных машин, однако оно не подходит для хранения важных данных.

Монтирование noasync

Наконец, рассмотрим комбинацию sync и async. Этот метод называется noasync и применяется во FreeBSD по умолчанию. При этом данные, имеющие отношение к индексным дескрипторам, записываются на диск синхронно, а фактические данные – асинхронно. В сочетании с soft updates (см. раздел «Soft Updates» далее в этой главе) noasync применяется для создания по-настоящему устойчивых систем.

Параметры монтирования FFS

В дополнение к типам монтирования система FreeBSD поддерживает несколько *параметров монтирования*. Необязательно знать подробности всех параметров монтирования, но надо, по крайней мере, знать об их существовании, чтобы применить в случае необходимости.

`noatime`

В FFS у каждого файла есть «отпечаток» времени доступа, называемый *atime*, в котором отражается время последнего доступа к файлу. Если у вас большое количество файлов и нет необходимости в таких «отпечатках», можно монтировать диск с параметром `noatime` и не обновлять информацию о времени доступа.

`noexec`

Запрещает выполнение любых программ в этом разделе. В разделе */home*, монтированном с параметром `noexec`, пользователи не смогут устанавливать и запускать свои программы, однако для полного предотвращения таких действий параметр `noexec` следует применять при монтировании */tmp*, */var/tmp* и других разделов, в которые пользователи могут записывать свои файлы.

`nosuid`

Предотвращает запуск setuid-программ на вашей системе. setuid-программы разрешают пользователям запускать программы от имени других пользователей. Например, некоторые программы (такие как login(1)) должны запускаться пользователем root, однако если программа login будет иметь статус setuid, выполнить ее сможет любой желающий.

Очевидно, setuid-программы должны разрабатываться тщательно. Необходимо предотвратить возможность их взлома и применения для неавторизованного доступа к системе. Вот почему многие администраторы обычно делают недоступными все ненужные setuid-программы. Для этого можно применить nosuid, однако это не принесет пользы при наличии оболочки сценариев, которая позволяет запускать сценарии как setuid. Примером такой оболочки является suidperl.

`nosymfollow`

Запрещает применять символические ссылки на файлы, или псевдонимы. (Так или иначе, символические ссылки применяются для создания псевдонимов файлов, расположенных в других разделах.) Псевдоним файла, который расположен на разделе, монтированном с параметром `nosymfollow`, создается при помощи жесткой ссылки (hard link), получающейся в результате вызова ln(1).

`nodev`

Запрещает применение любых файлов устройств, что помогает предотвратить ошибки при доступе к накопителю, используемому другой системой UNIX. Файловым системам, которым не положено иметь файлы устройств, их запрет ничем не повредит. В большинстве случаев файлы устройств должны быть лишь в корневом разделе (/).

При монтировании жесткого диска другой системы UNIX обязательно применяйте `nodev`. Случайный доступ к файлу устройств другой операционной системы наверняка приведет к аварии системы или разрушению данных!

В разделе, монтированном с параметром `nodev`, нельзя применять клетки, помните, что для клеток нужен минимальный набор файлов устройств.

Что монтировано сейчас?

Как определить, что монтировано на вашей системе? Можно начать с запуска mount(8) без аргументов. В результате будет выдан список всех монтированных файловых систем:

```
# mount
/dev/ad0s1a on / (ufs, local, soft-updates)

/dev/ad0s1f on /test1 (ufs, local, soft-updates)
/dev/ad0s1g on /test2 (ufs, local, soft-updates)
/dev/ad0s1h on /usr (ufs, local, soft-updates)
/dev/ad0s1e on /var (ufs, local, soft-updates)
procfs on /proc (procfs, local)
#
```

Здесь мы видим, что почти все файловые системы представляют разделы FFS, которые монтированы локально с включенными Soft Updates. (Soft updates рассматриваются далее в этой главе.) Если при монтировании разделов применяются такие функции, как SMB (см. главу 21), они будут показаны в выводе. Вызов mount – еще один быстрый способ получить имена устройств для каждого из разделов.

Как будет показано далее, с помощью mount можно решать и другие интересные задачи. Однако первым делом посмотрим, какие опасности подстерегают ваши разделы.

«Грязные» диски

Нет, диски не пачкаются при эксплуатации (хотя пыль на пластине причинит ей вред, да и вода, безусловно, будет некстати). «Грязный» дисковый раздел находится в неопределенном состоянии: операционная система запрашивает информацию, которая находится в этом разделе, но данные не были записаны полностью. Возможно, записана только часть блоков данных, либо изменен индексный дескриптор, но данные не записаны, либо эти два варианта скомбинированы.

Если при записи на диск выключается питание, в системе появляется «грязный» диск.

fsck

В состав FreeBSD входит мощный инструмент для проверки файловых систем – fsck(8). Если при загрузке системы будет найден «грязный» диск, fsck автоматически проверит диск и постарается устранить неполадки. Данные, не записанные на диск, будут утеряны, но fsck сделает все возможное, чтобы привести в порядок те данные, что остались. При благоприятном исходе все встанет на свои места – за исключением незаписанных данных.

Неудачный автоматический запуск fsck

Изредка перезагрузка заканчивается неудачно, и вы попадаете в однопользовательский режим, где вас просят запустить fsck вручную. Здесь у вас есть

выбор. Наберите `fsck`, и эта программа проверит каждый блок и индексный дескриптор на диске. Вероятно, она найдет все блоки, не связанные с соответствующими индексными дескрипторами. Далее программа постарается выяснить былое соответствие. Однако она не сможет сказать, к какому каталогу принадлежат эти файлы.

fsck спросит, надо ли выполнить перегруппировку. Если вы ответите «n», программа удалит поврежденные файлы. В случае ответа «y» потерянный файл будет помещен в каталог находок (например, */usr/lost+found*) в разделе, в котором они найдены. В качестве названия файла будет выступать некое число. С помощью grep пропавшие файлы можно поискать по контексту. Если файлов мало, их можно просмотреть с помощью more(1) и идентифицировать.

Отключение приглашения fsck

Если диск стал «грязным» во время масштабной операции, на нем может появиться множество потерянных файлов. Чтобы не тратить время, снова и снова набирая «y» и предписывая fsck восстанавливать каждый файл, можно запустить `fsck -y`, заранее ответив «y» на все вопросы. Это намного проще, чем сидеть и без конца набирать «y».

Примечание

Когда я работал с экспериментальными системами на -current, благодаря `fsck -y` в каталоге */usr/lost+found* иногда оказывалось содержимое всего диска. Восстанавливать файлы было трудно. А в системах FreeBSD-stable никаких трудностей с автоматическим ремонтом диска у меня не было.

Систему можно настроить на автоматический запуск `fsck -y` при начальной загрузке. Однако я не рекомендую такой шаг: если есть хотя бы малейший шанс, что моя файловая система вознесется в цифровые небеса, я хочу знать об этом. Я хочу сам набирать опасную команду и трепетать, наблюдая за работой fsck. Кроме того, всегда неприятно обнаружить «мусор» в системе, не имея ни малейшего понятия о том, что произошло. Однако если вы все же должны попробовать автоматический запуск, установите в */etc/rc.conf* следующий параметр:

```
fsck_y_enable="YES"
```

Что вместо fsck –y

Какой у вас есть выбор вместо `fsck -y`? Программы fsdb(8) и clri(8) позволяют отлаживать файловую систему и перенаправлять файлы в надлежащее место. Можно восстановить имена файлов и разместить их в исходных каталогах. Однако это трудно[1] – такой путь рекомендуется только Мастерам Файловых Систем из Тайного Клана Ниндзя.

[1] Здесь значение слова «трудно» аналогично его же значению во фразе: «Трудно покорить Эверест в сандалиях и шортах».

Монтирование и демонтирование дисков

Mount(8) монтирует файловые системы, размещенные на дисковых устройствах. (Перезагрузите машину FreeBSD, выберите однопользовательский режим (см. главу 3) и продолжайте чтение.)

При загрузке система монтирует корневой раздел, который предоставляет ей достаточно информации для базовой настройки. Основные сервисы системы будут запущены до монтирования остальных файловых систем. Но для того чтобы выполнить мало-мальски интересные операции в однопользовательском режиме, надо монтировать другие файловые системы.

Монтирование стандартных файловых систем

Для монтирования файловой системы, указанной в */etc/fstab*, например */usr* или */var*, введите такую команду:

```
# mount /usr
#
```

Раздел */usr* монтирован для чтения-записи.

Монтирование с параметрами

Для указания отдельного параметра, например «только для чтения», применяется аргумент `-o`. Мы уже говорили (в разделе «Типы монтирования FFS»), что к параметрам относятся `async`, `noasync`, `sync`, `rdonly`, `nodev` и `noexec`. Например, для монтирования */usr* «только для чтения» введите такую команду:

```
# mount -o rdonly /usr
#
```

Монтирование «только для чтения» – отличный выбор для поврежденных файловых систем. Мне приходилось иметь дело с настолько поврежденными файловыми системами, что они не поддавались программе fsck. (Это были серверы телеконференций с сотнями и тысячами статей. Обычно я что-нибудь с ними вытворяю – каждый должен как-то учиться.) Если вы хотите просто считывать информацию с диска без риска причинить ущерб данным, смонтируйте раздел «только для чтения» и копируйте с него данные сколько душе угодно. (Это не так в случае физического повреждения диска. Если одна из пластин распалась на части, вы обречены.)

Принудительное монтирование «чтение/запись»

Если раздел признан «грязным», то перед монтированием «чтение/запись» надо запустить fsck. Это не представляет трудностей, однако запуск fsck может быть неприемлем в силу различных причин. Если так, попробуйте применить ключ `-f` и выполнить принудительное монтирование. Однако такое монтирование с большой вероятностью приведет к аварии системы.

```
# mount -f /usr
#
```

Монтирование всех стандартных файловых систем

Для монтирования всех файловых систем, указанных в */etc/fstab*, применяется ключ `-a`. (К файловым системам будут применены ключи, заданные в */etc/fstab*.)

```
# mount -a
#
```

Монтирование в нестандартных каталогах

Чтобы монтировать файловые системы в произвольных каталогах, укажите имя устройства и точку монтирования. Такая возможность пригодится при установке нового диска, например для монтирования старых дисковых разделов */old/var*, */old/home* и т. д., чтобы можно было в любое время скопировать их содержимое.

```
# mount /dev/fd0 /home/mwlucas/floppy
#
```

Размонтирование

Завершив работу со съемным носителем, извлеките его с помощью umount(8). В качестве аргумента введите имя каталога, где монтирован носитель (*/home/mwlucas/floppy* в следующем примере).

```
# umount /home/mwlucas/floppy
#
```

Если вы не можете размонтировать устройство, то, возможно, к нему в данный момент осуществляется доступ. Например, вы не сможете демонтировать устройство, если приглашение на ввод команды выдается в каталоге носителя или на нем выполняется операция чтения или записи.

Soft Updates

Soft Updates – это самое значительное нововведение за всю историю FFS. Soft Updates организуют и приводят в порядок операции записи на диск. При этом метаданные файловой системы на диске остаются непротиворечивыми, производительность близка к производительности при асинхронном монтировании, а надежность соответствует надежности синхронного монтирования. Это не означает, что на диск будут записаны все данные – неполадки с электропитанием по-прежнему могут повредить данные. Однако Soft Updates предотвращают значительную часть трудностей.

Включение Soft Updates

Если рекомендации главы 1 выполнены, то система установлена с включенными Soft Updates. (Запуск mount без аргументов сообщит о том, включены ли Soft Updates.) Однако если Soft Updates не включены, то включить их (да и отключить) достаточно легко с помощью программы настройки файловых

систем tunefs(8). (tunefs имеет много других функций, однако для применения большей их части требуются глубокие знания файловой системы.)

Для включения Soft Updates загрузите систему в однопользовательском режиме, демонтируйте файловые системы, которые вы хотите изменить, и с помощью tunefs включите Soft Updates:

```
# umount /usr
# tunefs -n enable /usr
# mount /usr
```

Теперь загрузите систему в многопользовательском режиме – Soft Updates включены! Файловая система запомнит факт включения Soft Updates и будет активизировать их при последующих загрузках.

Чтобы отключить Soft Updates на том или ином разделе, замените в предыдущем примере слово «enable» на «disable».

Кэширование записи в дисках IDE и Soft Updates

Подобно файловым системам, Soft Updates лучше всего работают на жестких дисках SCSI в силу устойчивости архитектуры SCSI. Тем не менее Soft Updates также работают и на дисках с архитектурой IDE. Однако следует принять во внимание одно обстоятельство: многие современные диски IDE поддерживают *кэширование записи*.

Диски IDE, поддерживающие кэширование записи, имеют встроенную микросхему, которая запоминает данные, предназначенные для записи на диск. Такой механизм может не подойти для Soft Updates, поскольку Soft Updates уповают на «честность» жесткого диска – когда жесткий диск сообщает о записи данных на диск, Soft Updates предполагают, что данные находятся на пластине. Однако на самом деле механизм кэширования записи IDE сообщает о благополучном сохранении данных в дисковом кэше, а не о том, что данные записаны на диск. Фактическая запись данных на диск может произойти спустя секунду или более.

Этот разрыв во времени не опасен при единичных случаях его возникновения, однако на сервере это происходит постоянно. Поэтому, если вы беспокоитесь о своих данных, отключите кэширование записи, добавив в */boot/loader.conf* следующую строку:

```
hw.ata.wc=0
```

Отключение кэширования записи замедлит работу диска IDE, однако ваши данные будут в безопасности. Я благополучно применяю кэширование записи на настольном копьютере и лэптопе, где данные не записываются на диск постоянно, однако на сервере ни в коем случае нельзя оставлять кэширование записи IDE.[1]

---

[1] Вообще говоря, на сервере не следует устанавливать диски IDE, но я думаю, что к этому времени вы уже и так все поняли.

Кэширование каталогов в виртуальной памяти

Когда вы изучаете содержимое каталога (например, при запуске ls), системе нужно создать список файлов этого каталога. Подобная операция трудоемка. Если она выполняется часто, то производительность можно поднять за счет кэширования содержимого каталога. Такое кэширование включается с помощью sysctl `vfs.vmiodirenable`.

В новейших версиях FreeBSD этот параметр может быть установлен по умолчанию. Если нет, то автоматическое включение его при начальной загрузке обеспечивается с помощью следующей строки в */etc/sysctl.conf*:

```
vfs.vmiodirenable=1
```

Монтирование неродных файловых систем

Для наших целей любой диск, который не является разделом FFS, есть неродная (foreign) файловая система. К счастью, во FreeBSD реализована серьезная поддержка неродных файловых систем. Отметим, что будут работать только те функции, которые поддерживаются в самой файловой системе. (Например, FAT не поддерживает права доступа. Можно установить флаги безопасности на уровне файловой системы, однако это ничего не даст.)

Каждая файловая система имеет уникальную программу монтирования, которая учитывает все ее капризы, и каждой файловой системе нужна поддержка ядра. К счастью, программы монтирования автоматически загружают соответствующий модуль ядра.

Применение неродных монтированных систем

Для монтирования неродной файловой системы необходима та же информация, что и для монтирования FFS: имя устройства и точка монтирования. Также необходимо знать имя команды, которая монтирует файловые системы данного типа. (Различные файловые системы и команды для их монтирования будут рассмотрены в следующем разделе.)

Например, для монтирования CD-ROM надо указать */cdrom* как точку монтирования и имя устройства. В качестве примера выберем первый IDE-привод CD-ROM. По умолчанию это устройство */dev/acd0c*.

```
# mount_cd9660 /dev/acd0c /cdrom
#
```

Если CD-ROM монтирован, с него можно читать данные. Довольно легко, не правда ли?

При попытке монтировать диск командой, не поддерживаемой в данной файловой системе, будет выдано сообщение об ошибке. Я довольно часто сталкиваюсь с этим при монтировании незнакомых дискет:

```
# mount /dev/fd0 /mnt
mount: /dev/fd0 on /mnt: incorrect super block
#
```

Это дискета в формате MS-DOS, но все получится, если применить `mount_msdos`.

Независимо от типа монтированной файловой системы ее можно размонтировать с помощью umount(8):

```
# umount /cdrom
#
```

Типы неродных файловых систем

Вот список наиболее распространенных неродных файловых систем с кратким описанием и соответствующими командами монтирования.

MS-DOS

Во FreeBSD реализована значительная поддержка файловых систем FAT (DOS/Windows 9x File Allocation Table), обычно используемых на системах с двойной загрузкой и на дискетах.[1] Впрочем, вы можете фоматировать дискеты в FFS, поэтому не все из них имеют формат MS-DOS. Если попытка монтировать дискету как MS-DOS не удалась, попробуйте смонтировать ее как диск FFS. (Лично я применяю дискеты только для обмена файлами с машиной Windows, которой я не управляю, например при передаче файлов в типографию, и всегда форматирую дискеты как MS-DOS.)

Команда монтирования – mount_msdos(8).

Если вам приходится монтировать много устройств MS-DOS, ознакомьтесь как следует с каталогом */usr/ports/tools/mtools*. Здесь собраны программы для работы с MS-DOS, обладающие более высокой производительностью, чем инструменты FreeBSD.

NTFS

Стандартная файловая система Windows NT/2000/XP, NTFS, тесно интегрирована с ядром Microsoft. Чтобы записывать данные в раздел NTFS, необходимо глубоко понимать работу этой файловой системы. К сожалению, Microsoft не предоставляет такую информацию, поэтому вы можете считывать данные с раздела NTFS, а попытка записи может повредить этот раздел. Команда монтирования – mount_ntfs(8).

Примечание

Поскольку Microsoft так дорожит своим интерфейсом файловой системы и регулярно его изменяет, такие разделы нельзя считать надежными. Применение mount_ntfs может разрушить файловую систему.

ISO-9660

ISO-9660 – это стандартная файловая система для CD-ROM. FreeBSD позволяет считывать данные с CD-ROM, а также записывать их при наличии запи-

[1] Проще говоря, FAT используется везде, где есть DOS или Windows. – *Примеч. науч. ред.*

сывающего привода. Чуть ли не каждый CD-ROM, с которым вы столкнетесь, будет иметь формат ISO-9660. Команда монтирования – mount_cd9660(8).

Ext2fs

Стандартная файловая система Linux, ext2fs, поддерживает многие функции файловой системы FreeBSD, позволяя безопасно читать и записывать данные. Подобно монтированию NTFS, монтирование ext2fs довольно полезно при чрезвычайных обстоятельствах. Если веб-сервер на базе NT или Linux выйдет из строя, можно будет переставить его жесткий диск на рабочую машину FreeBSD и скопировать данные. Перенос физического жесткого диска – это не самый легкий способ передачи данных, однако при значительном повреждении машины этот способ самый быстрый. Для монтирования файловой системы ext2fs применяется команда mount_ext2fs(8).

Параметры монтирования и неродные файловые системы

Вместо того чтобы указывать различные команды монтирования для каждой файловой системы, можно задать тип файловой системы в качестве параметра mount(8). Для этого предназначен ключ `-t`. В таком способе монтирования нет особенных преимуществ и недостатков, но он действует.

```
# mount -t cd9660 /dev/acd0c /cdrom
```

Права доступа в файловых системах

Метод, применяемый для монтирования файловой системы, и пользователь, который ее монтирует, определяют права доступа монтированной файловой системы. Например, FFS и ext2fs хранят права доступа в файловой системе, ставя их в соответствие идентификаторам пользователей (UIDs). Поскольку ext2fs обычно работает подобно FFS, а вся информация о правах доступа находится в самой файловой системе, FreeBSD соблюдает эти права доступа.

Однако в NTFS своя система прав доступа. Поскольку NTFS лишь отдаленно похожа на UNIX, права доступа NTFS будут отброшены при монтировании этой файловой системы во FreeBSD. Она будет рассматриваться как аналогичная DOS или CD-ROM.

По умолчанию монтировать файловые системы может только root. Он же владеет всеми неродными файловыми системами. Если это не соответствует вашим предпочтениям, то с помощью ключей `-u` и `-g` можно указать ID пользователя и группы при монтировании файловых систем MS-DOS, NTFS или ISO-9660. Например, следующая команда подойдет при монтировании дискеты MS-DOS для пользователя «cstrzelc», причем этот пользователь сможет редактировать содержимое дискеты:

```
# mount_msdos -u cstrzelc -g cstrzelc /dev/fd0 /mnt
#
```

Теперь пользователь cstrzelc является владельцем файлов этой дискеты.

Примечание

Для того тобы разрешить пользователю монтировать файловые системы, присвойте sysctl `vfs.usermount` значение 1 и убедитесь, что пользователь владеет точкой монтирования.

Съемные носители и /etc/fstab

Диски CD-ROM традиционно монтируются в */cdrom*, а дискеты – в */mnt*. Можно упростить монтирование съемных носителей, добавив соответствующие записи в файл */etc/fstab*. Если файловая система имеет запись в */etc/fstab*, то при монтировании можно не указывать имя устройства. Другими словами, не надо помнить имена всех устройств и точные команды для монтирования тех или иных файловых систем.

```
# mount /mnt
#
```

Это проще, чем каждый раз набирать `mount_msdos /dev/fd0 /mnt`, не правда ли?

Указывая съемный носитель в */etc/fstab*, не забудьте добавить параметр `noauto`. В противном случае загрузка системы может остановиться в однопользовательском режиме, поскольку в накопителе на гибких дисках или в приводе CD-ROM может не оказаться носителя!

Создание дискеты

Процесс, который большинство пользователей Windows называют «форматированием дискеты», на самом деле является многоэтапным. Обычно он подразумевает форматирование диска, а также назначение метки и файловой системы. Во FreeBSD при создании дискеты надо выполнить все эти операции.

Примечание

Предположим, что есть гибкий диск емкостью 1,44 Мбайт, который был стандартом для аппаратных средств x86 свыше десяти лет. Если емкость диска нестандартная (800 Кбайт или какая-нибудь другая), потребуется внести поправки, однако последовательность шагов не изменится.

Низкоуровневое форматирование

При форматировании диска первым делом надо выполнить низкоуровневое форматирование с помощью fdformat(1). Эта программа принимает только два аргумента: размер дискеты и имя устройства.

```
# fdformat -f 1440 /dev/fd0
Format 1440K floppy `/dev/fd0.1440'? (y/n): y
```

Когда вы наберете y, fdformat начнет низкоуровневое форматирование, подготавливая диск к переносу на него файловой системы; файловая система

здесь не создается. Низкоуровневое форматирование – это самый медленный этап при подготовке дискеты.

Создание файловой системы FFS

Если вы создаете дискету FFS, то диску надо присвоить метку с помощью disklabel(8). Эта программа записывает на дискету базовую идентификационную информацию, устанавливает данные о разделах и даже может обозначить диск как загрузочный. Однако обозначение диска как загрузочного не подразумевает копирование на него необходимых программ; маркер, размещаемый на диске, нужен для того, чтобы системный BIOS воспринял этот диск как загрузочный.

Создание загрузочной дискеты FreeBSD – утомительный процесс, и тот, кому нужна такая дискета, может взять ее из установочного комплекта. В следующем примере мы добавим простую метку диска без каких бы то ни было специальных характеристик:

```
# disklabel -r -w /dev/fd0 fd1440
#
```

Ключ -r предписывает disklabel обратиться к диску на низком уровне. Это необходимо, поскольку на диске еще нет файловой системы. Ключ -w предписывает осуществить запись на диск: мы записываем в */dev/fd0* и устанавливаем стандартную метку для дискеты 1,44 Мбайт. (Полный список меток дискет можно найти в */etc/disktab*. Там же располагаются метки для устройств других типов.)

Наконец, создайте файловую систему с помощью newfs(8).

```
# newfs /dev/fd0
Warning: Block size restricts cylinders per group to 6.
Warning: 1216 sector(s) in last cylinder unallocated
/dev/fd0:       2880 sectors in 1 cylinders of 1 tracks, 4096 sectors
        1.4MB in 1 cyl groups (6 c/g, 12.00MB/g, 736 i/g)
super-block backups (for fsck -b #) at:
 32
#
(Предупреждение: размер блока ограничивает количество цилиндров в группе до 6.
Предупреждение: в последнем цилиндре не назначено 1216 секторов
/dev/fd0: 2880 секторов в 1 цилиндре 1 дорожки, 4096 секторов
1.4 Мбайт в 1 группах цилиндров (6 цилиндров на группу, 12.00 Мбайт в группе,
736 индексных дескрипторов в группе) резервная копия суперблока (для fsck -b)
в блоке номер 32.)
```

Создание файловой системы MS-DOS

Для перемещения данных между машинами Windows и FreeBSD требуется подготовить дискету в формате MS-DOS. Как и в предыдущем случае, запустите fdformat(8), однако дискете MS-DOS не надо присваивать метку.

```
# newfs_msdos /dev/fd0
/dev/fd0: 2840 sectors in 355 FAT12 clusters (4096 bytes/cluster)
```

```
bps=512 spc=8 res=1 nft=2 rde=512 sec=2880 mid=0xf0 spf=2 spt=18 hds=2 hid=0
#
```

Примечание

Для FFS вывод команды newfs выглядит более интересно, не правда ли? Причина заключается в том, что FFS – более сложная и интересная система по сравнению с MS-DOS. Впрочем, дискеты MS-DOS проще применять при наличии нескольких машин. Решите, что вам лучше подходит. При создании файловых систем можно использовать различные ключи; для получения подробной информации обратитесь к страницам руководства по newfs(8) или newfs_msdos(8).

Основы SCSI

По ходу книги я неоднократно повторял, что диски SCSI лучше дисков IDE. Это правда: диски SCSI быстрее, надежнее и дороже, чем диски IDE или EIDE. Кроме того, дисками SCSI труднее управлять, однако трудности проистекают из незнания технологии. Диски SCSI могут выполнять намного больше функций, чем диски IDE, что делает их более коварными.

Многие начинающие системные администраторы незнакомы с обслуживанием систем SCSI, поэтому часть времени я посвящу изложению необходимых сведений о SCSI.

Примечание

Если вы заинтересованы в глубоком изучении SCSI, обратитесь к книге Гэри Филда (Gary Field) «The Book of SCSI» (издательство No Starch Press).

Типы SCSI

Есть несколько типов SCSI: SCSI-1, SCSI-2, Wide, Ultra, LVD и т. д. Устройства SCSI кажутся похожими друг на друга, но посмотрев на разъемы, можно заметить отличия.

В устройствах SCSI-1 разъем 50-контактный, напоминающий кабель IDE. В более новых системах SCSI-2 разъем 68-контактный, с очень маленькими штырьками и трапециевидным[1] корпусом. Есть еще внешние системы SCSI, в которых применяются кабели других видов, например кабели Centronics, похожие на кабели старых параллельных принтеров. (Если вам предлагают кабель SCSI с 25 контактами, не берите его, даже с адаптером. Он снизит производительность системы, нарушит целостность данных и ваше душевное спокойствие.)

Адаптеры SCSI

Для применения SCSI нужен хост-адаптер. Это съемная карта, подобная видеокарте и другим картам периферийных устройств. Во многих серверах

[1] Часто такой разъем за его форму называют разъемом D-типа. – *Примеч. науч. ред.*

адаптеры SCSI установлены на системных платах, как адаптеры IDE в платах настольных компьютеров. Это упрощает управление устройствами.

Хост-адаптер SCSI – одно из наиболее вероятных узких мест в системе. Если диски быстрые, а адаптер медленный, то скорость передачи данных будет невелика.

Шины SCSI

Многие компоненты компьютера имеют *шину*, и SCSI не исключение. Если говорить упрощенно, то на шину устанавливаются устройства. На одной шине IDE есть два порта (разъемы на кабеле IDE), а на шине PCI – несколько (гнезда на материнской плате). Старые шины SCSI имеют 8 портов, включая порт, который задействован самим адаптером SCSI. Эти порты представляют собой разъемы, которые можно присоединить к кабелю SCSI. В старую систему SCSI можно вставить до 7 устройств. На более новой шине SCSI есть 16 портов (включая один порт для карты). Если в вашей системе достаточно места и разъемов питания,[1] можно вставить до 15 устройств.

В некоторых картах SCSI предусмотрено место для двух кабелей, поскольку на них есть две шины. В таких адаптерах шины имеют уникальные номера и рассматриваются по отдельности. (Если у вас одна шина, она тоже имеет уникальный номер – 0.)

Терминаторы и кабели

Сигналы SCSI передаются по всей длине кабеля. Во многих электротехнических системах (включая SCSI) сигнал, достигший конца провода, отражается и передается обратно. В случае SCSI в этом нет ничего хорошего, поэтому на конце шины SCSI надо поставить терминатор (маленькую деталь, похожую на колпачок), который предписывает сигналу остановиться и не отражаться.

В шинах SCSI терминатор должен стоять на каждом конце. В некоторых устройствах SCSI можно применять «внутренние терминаторы». Сигнал в шине SCSI останавливается на первом встретившемся терминаторе, поэтому при наличии «внутреннего терминатора» в середине шины некоторые компоненты не будут видны.

Может показаться, что найти и применять внешние терминаторы – дело хлопотное, но по возможности остановите свой выбор на них. Внешние терминаторы упрощают замену того или иного устройства.

Примечание

Не допускайте «Y-образного» размещения устройства SCSI на кабеле. Шина должна представлять собой прямую линию. Некоторые кабели якобы допускают такое размещение, но первое впечатление обманчиво – их нельзя применять таким образом.

---

[1] А также мощности блока питания. – *Примеч. науч. ред.*

Идентификаторы SCSI и LUN

Идентификатор (ID) SCSI – это адрес, который шина применяет для указания места назначения сигнала. Каждое устройство имеет уникальный SCSI ID, который идентифицирует его на шине. На физическом уровне шина маркирует каждый блок данных, указывая SCSI ID места назначения. Каждый SCSI ID должен быть уникален в большинстве систем; если устройства имеют идентичные номера, первое опрошенное при загрузке устройство захватит ID.

Каждый порт на шине имеет уникальный ID. Поскольку это компьютерная система, идентификаторы нумеруются, начиная с 0 и заканчивая 7 или 15 – в зависимости от шины. Типичный ID адаптера равен 7.

Некоторые системы SCSI также поддерживают номера логических устройств (Logical Unit Numbers LUNs). Благодаря продуманной схеме LUN несколько устройств могу разделять один идентификатор SCSI. Если каждый компонент в системе SCSI поддерживает LUN, эту схему можно использовать, причем устройства, разделяющие один SCSI ID, должны иметь уникальные LUN.

Управление системой SCSI аналогично отправке почты. Номер шины соответствует в этой аналогии названию города, а SCSI ID – названию улицы. Если на улице несколько зданий, почтальон опустит пакет в первом здании, которое попадется на его пути. Логический же номер, LUN, аналогичен номеру квартиры. (Пакеты не должны остаться в коридоре.) Если каждое устройство в системе имеет уникальный адрес SCSI, все будет замечательно. Если нет, то надо привести нумерацию в порядок.

FreeBSD и SCSI

Для того чтобы устройства SCSI правильно работали под управлением FreeBSD, необходимо внести изменения в ядро (см. главу 4). Есть две базовых задачи ядра, с которыми следует определиться: задержка SCSI при загрузке и привязка устройств.

Задержка при загрузке

Система SCSI содержит интеллектуальные микросхемы, обслуживающие поток данных, проходящий по шине. Их активизация при загрузке занимает больше времени по сравнению с аппаратными средствами IDE. По этой причине в процессе загрузки FreeBSD предусмотрена задержка – ожидание завершения активизации. Периодом задержки управляет параметр `SCSI_DELAY`:

```
options         SCSI_DELAY=8000
```

Время указывается в миллисекундах. Значение по умолчанию равно 15000 (или 15 секунд). Сам период времени зависит от аппаратных средств. Задержка в 15 секунд – это значение с запасом для старого оборудования. Для новых средств SCSI задержку можно снизить примерно до 5 секунд. Конечно, при наличии проблем можно вернуться к первоначальному значению. Задержка в 15 секунд ничему не повредит и не вызовет неполадок.

Привязка устройств

Другое неудобство заключается в том, что ядро FreeBSD нумерует устройства SCSI в порядке их обнаружения. (Каждая операционная система при загрузке интерпретирует SCSI каким-то образом; представленный вариант – это просто метод BSD.) В таком случае, изменяя расположение устройств на шине SCSI, можно менять порядок их опробования. Диск, который имел номер 0 при установке BSD, может стать диском 1 после добавления нового накопителя на магнитной ленте. Такое изменение может привести к тому, что разделы будут монтироваться в неверных точках монтирования.

С шинами SCSI могут быть похожие сложности. Если добавить еще одну шину SCSI, шины будут перенумерованы! Можете себе представить абсолютный хаос, вызванный любой из этих проблем.

Во избежание этих трудностей данную информацию можно жестко закодировать в ядре. Процесс, избавляющий от подобной путаницы, называется *привязкой* (*wiring down*) устройств SCSI. Чтобы выполнить привязку устройств, надо знать SCSI ID, номер шины SCSI и LUN (если применяется) каждого устройства в цепочке SCSI, доступных в */var/run/dmesg.boot*.

Например, в моей тестовой системе есть следующие записи dmesg для адаптера SCSI:

```
ahc0: <Adaptec aic7880 Ultra SCSI adapter> port 0xac00-0xacff mem 0xd9000000-
0xd9000fff irq 11 at device 10.0 on pci0
aic7880: Ultra Wide Channel A, SCSI Id=7, 16/255 SCBs
```

В первой строке сообщается, что основная карта SCSI – это адаптер «Adaptec aic7880 Ultra». Во второй содержится дополнительная информация об адаптере на этой карте. На самом деле здесь есть только одна физическая карта. (Изучая файлы *dmesg*, вы многое узнали о компонентах вашего компьютера, не правда ли?) Хост-адаптер использует SCSI ID 7. LUN не применяется. Сняв корпус и взглянув на физическую карту, я могу убедиться, что есть только одна шина SCSI.

Далее в файле *dmesg* есть записи для дисков:

```
...
❶ da0 at ❷ ahc0 ❸ bus 0 ❹ target 2 ❺ lun 0
...
❻ da1 at ahc0 bus 0 target 8 lun 0
...
```

Эти строки сообщают, что диск da0 (❶) подсоединен к карте SCSI ahc0 (❷). Также ясно, что карта располагается на шине 0 (❸). Слово «target» – это SCSI ID 2 (❹). Наконец, LUN равен 0 (❺). Второй диск SCSI (❻) расположен на той же самой карте SCSI. Та же шина и тот же LUN. Однако SCSI ID равен 8.

Для того чтобы выполнить привязку дисков, сообщите ядру расположение каждого из них. Тогда ядру не придется задумываться о способе нумерации устройств SCSI:

```
device        da0 at scbus0 target 2 unit 0
device        da1 at scbus0 target 8 unit 0
```

Аналогично можно поставить в соответствие определенной шине SCSI ту или иную карту и даже определенное гнездо на карте. Например, мы знаем, что шина SCSI 0 находится на ahc0. Сейчас в данной системе есть только одна карта SCSI, однако мы не хотим, чтобы новая карта SCSI нарушила работу шины SCSI 0. Мы «привяжем» шину SCSI к определенному гнезду на карте, задав следующую конфигурацию ядра:

```
device        scbus0 at ahc0 bus 0
```

После введения параметров пересоберите ядро и перезагрузите систему. Устройства будут сконфигурированы и поименованы согласно внесенным изменениям. В этом нет ничего примечательного, ибо это то, чего вы ожидали.

Для того чтобы убедиться в привязке устройств, поэкспериментируйте с конфигурацией ядра. Скажем, откройте конфигурационный файл и измените все экземпляры da0 на da7. После перекомпиляции и перезагрузки устройства */dev/da1* и */dev/da7* появятся в загрузочных сообщениях. (Этот маленький тест может затруднить загрузку системы. Не меняйте от нечего делать имена дисков, на которых располагаются корневой раздел (root) и раздел */usr*!)

Добавление новых жестких дисков

Добавление жестких дисков может оказаться трудным делом: диски надо отформатировать, разместить на них файловые системы, смонтировать и перенести на них данные. Упростить первые шаги можно двумя способами: либо применить sysinstall(8), простой и быстрый, либо выполнять все операции в командной строке. Если вы спешите, то наверняка выберете sysinstall.

Изредка при работе с sysinstall возникают трудности со старыми дисками (емкостью 300 Мбайт и менее). У того, кто устанавливает старые диски, может не остаться другого варианта, кроме метода командной строки. В этом случае на помощь придет Руководство по FreeBSD (FreeBSD Handbook).

Предположим, что вы добавляете диск в существующую систему, а ваша конечная цель состоит в том, чтобы перенести на этот диск данные. Рассмотрим два примера: создание */usr/obj* на этом диске и перемещение на него */home*.

Примечание

До начала любых операций с дисками выполните полное резервирвное копирование системы. Одна глупая опечатка при наборе команд может разрушить систему! Например, можно случайно переформатировать корневую файловую систему.

Создание участков

В первую очередь при работе с новым жестким диском следует создать на нем разделы. Выполните следующие шаги:

1. Войдите в систему как root и запустите sysinstall. Мы будем конфигурировать систему уже после установки, поэтому выбираем Configure и начинаем с Fdisk.
2. Меню покажется знакомым; вы применяли его при установке FreeBSD. (См. снимки экранов на рис. 1.4 главы 1.) Вы увидите существующий диск FreeBSD и новый диск. Выберите новый диск.
3. Если диск взят с другого сервера, на нем может оказаться файловая система. Можно сохранить содержимое диска, а можно очистить его и начать с нуля. Обычно проще всего удалить существующие разделы и файловые системы. С помощью клавиш-стрелок перейдите к существующему разделу и нажмите клавишу D для его удаления.
4. Далее создайте новый участок, нажав клавишу C, или задействуйте весь диск (клавиша A). На сервере лучше задействовать весь диск. Выбрав конфигурацию участков, сохраните внесенные изменения нажатием клавиши W. Появится следующее предостережение:

```
WARNING:  This should only be used when modifying an EXISTING installation. If
you are installing FreeBSD for the first time then you should simply type Q when
you're finished here and your changes will be committed in one batch
automatically at the end of these questions.  If you're adding a disk, you should
NOT write from this screen, you should do it from the label editor.
Are you absolutely sure you want to do this now?
(ПРЕДОСТЕРЕЖЕНИЕ: Такой вариант следует применять только при изменении
СУЩЕСТВУЮЩЕЙ системы. Если вы устанавливаете FreeBSD с нуля, то после внесения
изменений нажмите <Q>. В этом случае все изменения будут одновременно приняты
после получения ответов на данные вопросы. Если вы добавляете диск, вам
НЕ следует осуществлять запись с этого экрана. Нужно перейти в редактор меток.
Абсолютно ли вы уверены в своем выборе?)
```

5. Да, вы абсолютно уверены. Перейдите к пункту Yes и нажмите Enter.
6. Далее будет задан вопрос о необходимости установки на этот диск менеджера загрузки. На дополнительном диске он не нужен, поэтому спускаемся к пункту Standard и нажимаем Space. Затем спускаемся к пункту OK и нажимаем Enter. Программа sysinstall должна сообщить, что информация FDISK записана. Теперь на диске есть участок FreeBSD. Выйдем из этого экрана systinstall и перейдем к созданию разделов на участке.

Создание разделов

Для создания разделов на диске выполните следующие шаги:

1. Выберите пункт Label в sysinstall. С помощью команды c здесь можно создать новый раздел, указав его размер в мегабайтах, гигабайтах, дисковых блоках или дисковых цилиндрах. (Обычно размер указывают в гигабайтах или мегабайтах.) Кроме того, необходимо определиться с предназначением каждого нового раздела – будет ли это файловая система или пространство свопа.

Примечание

Создавая разделы, убедитесь, что в верхней части экрана расположено имя нового диска. Вы же не собираетесь переразмечать текущий диск!

2. Включите Soft Updates, если они не включены по умолчанию.
3. Назначив метки для всех разделов нового диска, снова нажмите клавишу W для записи изменений на диск. Теперь у вас есть таблица разделов. (Обратите внимание на имена разделов (например, da7s1e); они понадобятся далее.)
4. Наконец, нажмите W для подтверждения изменений. Должно появиться текстовое окно, подобное окну newfs. (Процесс может затянуться на несколько минут; форматирование жесткого диска 100 Гбайт – непростая задача!)
5. По завершении процесса выйдите из sysinstall.

Конфигурирование /etc/fstab

Если вы добавляете пространство свопа, то надо сконфигурировать данный раздел в */etc/fstab*, чтобы система его опознавала. (Вы записали имена новых разделов, не так ли?) Вероятно, имя раздела свопа примерно такое: da7s1b (замените da7 на имя вашего диска). В */etc/fstab* уже есть запись о существующем пространстве свопа. Эту запись можно применить в качестве шаблона при добавлении нового раздела свопа.

Например, пусть тестовому серверу соответствует такая запись о свопе:

```
/dev/da0s1b          none          swap     sw              0        0
```

Если имя нового диска – */dev/da7*, а раздел свопа создан на */dev/da7s1b*, добавьте такую строку:

```
/dev/da7s1b          none          swap     sw              0        0
```

Во время следующей загрузки система FreeBSD найдет эту запись и активизирует новое пространство свопа.

Кроме того, похожую запись следует добавить для нового раздела данных. Вот запись в */etc/fstab*, монтирующая раздел */dev/da7s1e* в */crud* – новой точке монтирования, созданной для этого диска:

```
/dev/da7s1e            /crud           ufs      rw              0        0
```

Перенос существующих файлов на новые диски

Вероятно, новый диск призван заменить существующий раздел FreeBSD либо взять на себя часть данных существующего раздела. Для этого надо предоставить системе доступ к новому разделу во временной точке монтирования. Переместите файлы из старого раздела в новый. Затем заново смонтируйте раздел в требуемом месте.

Временное монтирование

Предположим, у вас есть новый раздел */dev/da7s1e*, который вы хотите задействовать для */usr/src*, переместив файлы из существующего */usr/src* в новый раздел. Для того чтобы можно было обращаться к новому разделу, его надо временно смонтировать в каком-нибудь другом месте.

Временным пристанищем может быть любой каталог вашей системы. По традиции в качестве временной точки монтирования разделов применяется каталог */mnt*:

```
# mount /dev/da7s1e /mnt
#
```

Перемещение файлов

Теперь надо перенести файлы из их текущего местоположения в новый раздел, не изменяя права доступа к файлам и не воздействуя на них любым другим образом. Такая задача решается довольно просто с помощью tar(1). (Программа tar применяется для сохранения прав доступа к файлам.)

С помощью tar существующие данные можно сохранить на ленте или в файле, а затем разархивировать их в новом месте. Однако последовательно выполнять эти операции неудобно. Объединив команды tar, можно обойтись без промежуточного этапа:

```
(cd /old/directory && tar cf - .) | (cd /newplace && tar xpf -)
```

Эта строка может озадачить тех, кто не говорит на языке UNIX на вечеринках. Разберем ее по частям. Во-первых, вы переходите в старый каталог и архивируете все его содержимое с помощью `(cd /old/directory && tar cf - .)`. Затем вывод этой команды переносится в новый каталог, где данные разархивируются с помощью `| (cd /newplace && tar xpf -)`. Подождите несколько минут, и дело будет сделано.

Например, для перемещения */usr/src* на новый раздел, временно монтированный в */mnt*, подойдет такая последовательность команд:

```
# ( cd /usr/src && tar cf - . ) | ( cd /mnt && tar xpf - )
```

Она не проста, но недоступной пониманию ее не назовешь. Конечно, кому-то будет удобнее создать tarball в одном месте и разархивировать его в другом. Все, что для этого потребуется – дисковое пространство.

Примечание

Такая последовательность команд не удалит файлы в исходном каталоге. Убедившись, что копирование прошло успешно, можете удалить файлы командой `rm -rf`.

Перенос активных файлов

Вам не удастся благополучно переместить файлы, если они в этот момент подвергаются изменению. Например, при переносе почтового спула на но-

вый раздел выключите почтовые сервисы. В противном случае файлы изменятся, когда вы будете их копировать.

Составное монтирование

Предположим, что вам нет дела до старых данных и вы просто хотите освободить место на диске. Вы планируете восстановить свои данные с резервной копии. Разумно.

Все файловые системы BSD – *составные* (*stackable*). Они обладают расширенными возможностями, которые не особо полезны в повседневном администрировании. Однако эти возможности придут на помощь, когда потребуется распределить один раздел между двумя дисками.

Предположим, что ваши данные хранятся в */usr/src*. (Так оно и есть, если вы следовали советам по установке или обновили систему.) Выполните команду ls и убедитесь, что данные находятся именно там. Теперь смонтируйте новый раздел в */usr/src*:

```
# mount /dev/da7s1e /usr/src
#
```

Если вы зайдете в каталог */usr/src*, то увидите, что он пуст.

Есть одна загвоздка: новый раздел монтирован «поверх» старого диска, а все данные по-прежнему находятся на старом диске. Если размонтировать новый раздел и снова проверить каталог, вы увидите, что данные чудесным образом восстановлены! Новый раздел скрывал старый.

Вы не можете это увидеть, но данные на старом диске по-прежнему занимают определенное пространство. (Объем задействованного дискового пространства можно выяснить с помощью df(1) и du(1).) Если вы перераспределяете диск для получения дополнительного пространства, а новый диск монтируете поверх старого, дисковое пространство на исходном диске будет недоступно.

Мораль: даже при восстановлении данных с резервной копии надо убедиться, что данные перемещены с исходного диска и дисковое пространство освобождено.

17

RAID

Одна из важных особенностей систем SCSI в том, что они позволяют применять RAID, или Redundant Array of Independent Disks (массив независимых дисковых накопителей с избыточностью).[1] Раньше символ «I» в аббревиатуре «RAID» означал «Inexpensive» (недорогой), но это определение относительно. Массив RAID емкостью в один терабайт стоит намного меньше отдельного диска емкостью в один терабайт, но, тем не менее, он очень дорог.

Системы RAID распределяют данные между дисками для улучшения производительности и надежности. Существуют программные и аппаратные реализации RAID.

Сравнение программных и аппаратных RAID

FreeBSD поддерживает аппаратные и программные RAID. Аппаратным массивом RAID управляет контроллер SCSI. Хост-адаптеры, способные обслуживать RAID, называются *RAID-контроллерами*. В аппаратной реализации RAID контроллер выполняет все вычисления, связанные с размещением данных на жестких дисках, тем самым снижая нагрузку на систему. Большинство аппаратных RAID очень надежны, а аппаратный контроллер – это, бесспорно, лучший способ обслуживания RAID.

Программными RAID управляет операционная система. Именно ОС приходится управлять распределением данных по дискам. Этот метод увеличивает нагрузку на систему, однако обходится дешевле аппаратного RAID.

Применять аппаратный RAID намного легче, чем программный, поскольку обычно в первом случае достаточно следовать инструкциям в документации. Как правило, в аппаратных RAID есть BIOS, управляемый меню. BIOS позволяет назначать размеры разделов и восстанавливать поврежденные диски. И это все, что вам надо знать. Однако применение программных RAID подразумевает, что системные администраторы отдают себе отчет в своих

---

[1] Также существуют системы RAID с интерфейсом IDE, однако они страдают многими недостатками IDE (подробно описанными в главе 1).

действиях. В этой главе будут подробно рассмотрены программные RAID. В основе аппаратных RAID лежит та же теория, однако кроме нее и пунктов меню обсуждать в аппаратных RAID почти нечего.

Уровни RAID

RAID бывают разных типов. Наиболее популярны RAID-0, RAID-1 и RAID-5.

RAID-0 чаще всего называют *разбивкой на блоки* (*striping*). Формально данный массив – вообще не RAID. Он требует наличия по крайней мере двух дисков, данные между которыми распределяются таким образом, что повышаются производительность и общая емкость массива, однако данные сохраняются без избыточности. RAID-0 можно применять для объединения нескольких дисков по 100 Гбайт и создания одного крупного виртуального диска. Однако в случае сбоя и разрушения данных на одном диске вся система, в которую входит этот диск, станет бесполезной. Для доступа к информации потребуется восстанавливать данные с резервной копии. RAID-0 полезен в тех случаях, когда нужна одна большая-большая файловая система, не дающая выигрыша в надежности.

RAID-1 называют *зеркалированием* (*mirroring*). Содержимое одного диска дублируется на другом. (Количество дисков должно быть кратно двум.) Это хороший метод, который при невысокой стоимости предоставляет надежную схему хранения данных.

В RAID-5 данные по частям дублируются на всех дисках. Они организованы таким образом, что потеря одного диска не разрушит данные. В системах, допускающих «горячую замену», поврежденный диск можно заменить и ввести в строй даже во время работы системы. Повторю: нужны два диска и более.

Наконец, существует RAID-10 (известный еще как RAID-0+1). Это комбинация разбивки на блоки и зеркалирования. Для реализации такой схемы необходимы как минимум четыре диска. Число дисков должно быть кратно двум. Диски распределяются по двум наборам. Каждый набор дисков разбивается на блоки. Каждый набор блоков зеркалирует другой набор. Если такая схема вам по карману, применяйте именно ее.

При наличии выбора лучше остановиться на аппаратном RAID, поскольку его легче обслуживать. Кроме того, его реализация требует меньших навыков. Если у вас достаточно дисков, выберите RAID-10. Если дисков мало, создайте RAID-5 или RAID-1 (в таком порядке). (В домашней системе я применял RAID-0, но не рекомендую такую схему для серьезных приложений.)

Программные RAID

Обратимся к программным RAID, поскольку они намного более сложны, намного больше утомляют, а запускать их значительно труднее, нежели аппаратные RAID. В конце концов, пользователям с ограниченными средствами часто приходится прибегать к программным RAID. Если вы применяете ап-

паратные RAID, прочтите эту главу для ознакомления с общими понятиями, а затем обратитесь к своему руководству по аппаратному RAID и выясните, какие клавиши надо нажимать.

В состав FreeBSD входят два менеджера программных RAID: ccd и Vinum. ccd намного старше. Этот пакет может обслуживать только RAID-1. Vinum – более новая, быстрая и мощная программа. Она обслуживает RAID-0, 1, 5 и 10. Программа Vinum способна давать отличные результаты при низких затратах, поэтому мы будем обсуждать именно ее.

Примечание

Не существует операционных систем с программным RAID для слабонервных и для тех, кто не готов разбираться с проблемами. Это относится и к FreeBSD.

Дисковые компоненты Vinum

Vinum разделяет диски по трем отдельным типам: тома (volumes), накопители (drives) и поддиски (subdisks).

Тома

Vinum создает виртуальные диски, или *тома*. Один том Vinum похож на стандартный раздел диска. На нем можно осуществлять считывание и запись, его можно форматировать и т. д. Пользователи и программы на самом деле видят том, а не дисковый раздел. На одном разделе нельзя смешивать данные Vinum и стандартные файловые системы.

Накопители

Разделы, предназначенные для Vinum, называются *накопителями* (*drives*). В одном томе Vinum может быть сколько угодно накопителей. (Для Vinum можно задействовать и один диск, однако в этом случае придется столкнуться со значительными издержками программных RAID, не получив никакой пользы.)

При желании под накопители Vinum можно выделить целые диски, однако поступать так необязательно. Зачастую внешний раздел жесткого диска я предусматриваю для пространства свопа, а оставшуюся часть выделяю для накопителя Vinum.

Поддиски

Накопители Vinum содержат *поддиски*. Они представляют собой области, отведенные для Vinum на определенном накопителе Vinum. (Напомним, что в Vinum накопитель представляет собой раздел – пространство, предназначенное для данных Vinum в разделе, выделенном для Vinum.) При желании можно даже создать два поддиска на одном физическом диске, разместив их в разных разделах. (В этом нет особой пользы. Разве что таким образом можно продемонстрировать плохую работу программных RAID при плохой настройке.)

Итак, том создается из накопителей. Накопители содержат поддиски.

Типы плексов Vinum

Плекс (*plex*) – это представление всех данных в томе. Пока у вас есть хотя бы один полный плекс, все ваши данные доступны. Зеркалирование подразумевает наличие более одного плекса в томе. Vinum поддерживает до восьми плексов на том, поэтому данные можно хранить со значительной избыточностью. Способ организации плексов определяет вид RAID, который будет применяться.

Объединенный плекс

Для организации одного большого тома создайте *объединенный плекс* (*concatenated plex*), который последовательно охватит все доступные поддиски. Таким образом будет получен только один плекс, или одно представление данных.

Объединенный плекс – это наименее эффективный способ применения дисковых ресурсов. Этого вполне достаточно для двух дисков IDE на одном контроллере, но в случае SCSI такая схема будет пустой тратой системных ресурсов. Кроме того, это даже не RAID; это просто большой виртуальный диск. Однако для зеркалирования можно применить два объединенных плекса, а тома, построенные из объединенных плексов, можно легко расширить, установив дополнительные диски.

Плекс с разбивкой на блоки

Плекс с разбивкой на блоки представляет собой массив RAID-0, в котором данные распределены по дискам с целью увеличения производительности. Поддиски в таком плексе должны иметь одинаковый размер, а их количество не должно быть меньше двух.

Один плекс, разбитый на блоки, не обеспечивает избыточности данных. Однако можно построить два таких плекса на четырех дисках и создать RAID-0 с избыточностью. Если вы рассматриваете такую возможность, то лучше обратите внимание на RAID-5.

Зеркалированный плекс

В *зеркалированном плексе* данные воспроизводятся на нескольких плексах. Напомним, что поддиски в зеркалированном плексе должны иметь одинаковый размер. Для зеркала необходимо по меньшей мере два плекса, следовательно, здесь действительно есть избыточность данных. Это RAID-1.

Плекс RAID–5

Плекс RAID-5 разбивает на блоки и зеркалирует данные на нескольких поддисках. На нескольких дисках с достаточной избыточностью хранятся множественные экземпляры данных. Для организации RAID-5 требуется не меньше трех поддисков одного размера.

Примечание

Мы не будем подробно рассматривать Vinum RAID-5. Примеры и описание RAID-5 или RAID-10 можно найти на веб-сайте Vinum по адресу *http://www.vinumvm.org*. Прежде чем начинать работу с такими массивами, необходимо понять основы, изложенные в данном разделе.

Подготовка накопителей Vinum

На примере тестовой системы рассмотрим этапы настройки объединенного плекса, плекса с разбивкой на блоки и зеркалированного плекса. Во всех случаях необходимо выполнить следующее:

- Выбрать разделы, которые станут накопителями.
- Создать поддиски на этих накопителях.
- Сконфигурировать Vinum для создания соответствующих плексов на базе этих накопителей.

Все действия будем выполнять на одной и той же тестовой системе, в которой изначально представлены следующие диски и разделы:

```
# df
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad4s1a    248111    73649   154614    32%    /
devfs               1        1        0   100%    /dev
/dev/ad4s1f   2032839   133492  1736720     7%    /test1
/dev/ad4s1g   2032839  1266476   603736    68%    /test2
/dev/ad4s1h  29497862  3975801 23162233    15%    /usr
/dev/ad4s1e   3048830   241823  2563101     9%    /var
procfs              4        4        0   100%    /proc
/dev/da0s1e   3525041        1  3243037     0%    /crud
/dev/da1s1e   3758456        1  3457779     0%    /crud2
#
```

df(1) показывает объем используемого дискового пространства и точки монтирования разделов. Как видно из предыдущего вывода, в системе есть два пустых раздела на дисках SCSI – */crud* и */crud2*. (Идентифицировать диски как SCSI можно по именам устройств в начале строк. О незаполненности разделов можно судить по значениям `0%` в колонке `Capacity`.) Данные разделы будут задействованы для организации виртуального диска, разбитого на блоки. Я уже сохранил все данные этих разделов. Это необходимо, поскольку процесс настройки Vinum полностью перезаписывает содержимое накопителя!

Для подготовки накопителей Vinum первым делом необходимо размонтировать разделы:

```
# umount /crud
# umount /crud2
#
```

Повторный запуск df должен показать, что накопители уже невидимы. Удалите из */etc/fstab* записи, соответствующие этим разделам, чтобы система

FreeBSD не пыталась вернуть их на место при начальной загрузке. Теперь все готово для следующего шага.

Выделение разделов для Vinum

Итак, в системе есть незанятые и немонтированные разделы, и можно выделить их для Vinum, воспользовавшись disklabel(8). Эта программа преобразует стандартный раздел в накопитель Vinum. Она считывает и записывает «метки» на диске, выдавая базовую информацию о начале и конце раздела, их форматировании и физических характеристиках. Команда `disklabel diskname` выдаст существующую метку диска. Один из дисков Vinum в нашем примере – */dev/da1*, поэтому рассмотрим следующий вывод:

```
# disklabel da1
# /dev/da1c:
type: SCSI
disk: da1s1
...
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  b:  1048576        0      swap                        # (Cyl.    0 - 65*)
  c:  8803557        0    unused        0     0         # (Cyl.    0 - 547*)
  e:  7754981  1048576    4.2BSD     1024  8192    22   # (Cyl.   65*- 547*)
#
```

disklabel порождает большой объем вывода, включая информацию о многих параметрах, которые нельзя изменить (о скорости диска, количестве цилиндров и т. д.). В наших примерах эта информация отброшена, однако вы можете запустить disklabel и все увидеть своими глазами.

Сейчас нам нужны сведения, представленные в конце вывода disklabel, если говорить точнее, то мы хотим знать, где размещены разделы.[1] df показал, что */crud* на самом деле соответствует */dev/da0s1e*. Предыдущий вывод disklabel – для диска da1s1. Строка "e" в таблице разделов представляет раздел, который будет выделен для Vinum:

```
  e:  7754981  1048576    4.2BSD     1024  8192    22   # (Cyl.   65*- 547*)
```

Запуск `disklabel -e` откроет карту диска (disklabel) в текстовом редакторе. Вы можете изменить колонку `fstype` в записи участка с `4.2BSD` на `vinum`. После сохранения и выхода из редактора измененная информация disklabel будет записана на диск. После этого снова запустите `disklabel da0` и проверьте свою работу. Таблица разделов должна выглядеть примерно так:

```
#        size   offset    fstype   [fsize bsize bps/cpg]
  b:  1048576        0      swap                        # (Cyl.    0 - 65*)
  c:  8803557        0    unused        0     0         # (Cyl.    0 - 547*)
  e:  7754981  1048576     vinum                        # (Cyl.   65*- 547*)
```

[1] Возможно, вы обратили внимание, что disklabel сообщает о наличии восьми разделов, хотя мы видим только три. Участок может поддерживать до восьми разделов, однако в данном примере пять разделов фактически не используются.

Таким же образом отредактируйте метку диска раздела Vinum на другом диске. Теперь дисковые разделы выделены для Vinum.

Конфигурирование Vinum

Дисковое пространство для Vinum выделено, и его можно конфигурировать. Если задействовать встроенные инструменты, то базовая настройка будет простой.

В состав Vinum входит программа vinum(8), предназначенная для взаимодействия с модулем ядра, обслуживающим Vinum. Если ввести `vinum` в командной строке, то вы окажетесь в интерактивной оболочке Vinum, где можно регулировать конфигурацию, просматривать настройки и выполнять любые операции Vinum. В качестве альтернативы можно просто задавать надлежащие аргументы в командной строке. В этом случае vinum выполнит одно действие и завершит работу. Рассмотрим как метод командной строки, так и интерактивный режим.

Примечание

Совпадение имен программы и функции может внести путаницу. Когда речь пойдет о всей системе Vinum, первая буква имени будет большой (V). Программа описана в разделе руководства (vinum(8)).

vinum.conf

Свою фактическую конфигурацию Vinum хранит в базе данных на диске. Ее нельзя прочесть без применения vinum(8). Но можно создать конфигурационный файл vinum(8), который пригодится при первоначальной настройке Vinum. В нем содержатся все параметры, он облегчает проверку первоначальной настройки, а также изменение и пересборку конфигурации.

Однако после настройки Vinum этот файл теряет свое значение, так как фактическая конфигурация хранится в базе данных на диске. Изменения в систему Vinum можно вносить напрямую, не прибегая к файлу настройки. Он не содержит текущих настроек, для выяснения которых надо запустить vinum(8).

Примечание

В этой главе файл */etc/vinum.conf* будет выступать в качестве файла конфигурации. Обнаружив такой файл в своей системе, помните, что он может не играть никакой роли.

Объединенный плекс

Начнем с настройки отдельного объединенного тома, чтобы получить максимально возможное дисковое пространство. Организуем один плекс. Другими словами, у нас будет только один экземпляр данных. В этой модели нет избыточности, однако она позволяет получить большее дисковое пространство.

Вот файл */etc/vinum.conf* для объединенного плекса:

```
drive alpha device /dev/da0s1e
drive beta device /dev/da1s1e
volume test
      plex org concat
      sd length 3243037k drive alpha
      sd length 3457779k drive beta
```

Обратите внимание на первые две строки листинга. Прежде всего необходимо идентифицировать разделы с помощью ключевого слова `drive`. В этом примере есть два накопителя – */dev/da0s1e* и */dev/da1s1e*. (Напомним, что по терминологии Vinum разделы, выделенные для Vinum, являются *накопителями*.) Каждому накопителю необходимо уникальное имя. Я назвал их *alpha* и *beta*.

Далее с помощью ключевого слова `volume` надо задать имя для тома, или виртуального раздела. В данном случае в третьей строке листинга я назвал его *test*. Затем в четвертой строке с помощью ключевого слова `plex org` мы задаем вариант организации плекса. Слово `concat` указывает на объединение (concatenation).

Наконец, в пятой и шестой строках мы указываем Vinum размеры поддисков этого накопителя. Напомним: первоначальный запуск df показал, что на */dev/da0s1e* было доступно 3 243 037 Кбайт, а на */dev/da1s1e* – 3 457 779 Кбайт. Все доступное пространство на этих разделах можно задействовать для Vinum.

Примечание

Настройки тома могут быть более тонкими и задавать вид дисковых блоков, их размер и т. п., однако такой уровень детализации для Vinum необязателен. По умолчанию в Vinum предусмотрены разумные значения параметров.

Создание тома

Теперь с помощью `vinum create` создадим том, указав в качестве аргумента имя конфигурационного файла:

```
# vinum create /etc/vinum.conf
❶ 2 drives:
D alpha                 State: up       /dev/da0s1e     A: 384/3551 MB (10%)
D beta                  State: up       /dev/da1s1e     A: 409/3786 MB (10%)

❷ 1 volumes:
V test                  State: up       Plexes:       1 Size:       6543 MB

❸ 1 plexes:
P test.p0             C State: up       Subdisks:     2 Size:       6543 MB

❹ 2 subdisks:
S test.p0.s0            State: up       D: alpha        Size:       3167 MB
S test.p0.s1            State: up       D: beta         Size:       3376 MB
#
```

Внимательно изучите вывод vinum(8) и убедитесь, что результаты соответствуют значениям, заданным в конфигурационном файле. Наиболее важный

показатель верного результата – состояние up в каждой записи, что означает успешность операции.

Рассмотрим вывод этой команды. Прежде всего, есть два накопителя, alpha и beta, которые находятся в состоянии up (❶). В этих же строках есть имена устройств и объем занятого пространства.

Далее идет список томов (❷). У нас есть один том с именем test и размером 6543 Мбайт. Аналогично, есть один плекс с двумя поддисками (❸). Плекс назван по имени тома, к которому добавлены замыкающие символы ".p0".

Наконец, мы видим поддиски и накопители, на которых были созданы эти поддиски (❹). Каждый поддиск назван по имени тома, назначенного плексу. К имени тома добавляется произвольный номер поддиска, например test.p0.s1.

Теперь раздел Vinum доступен. Для него в */dev/vinum* определено имя устройства, соответствующее имени тома. Например */dev/vinum/test* для тома *test*.

Инициализация разделов Vinum

Прежде чем задействовать том Vinum, надо создать файловую систему. Поскольку sysinstall не распознает разделы Vinum (пока!), данная операция выполняется с помощью newfs(8). Если вы опытный системный администратор, то можете задать ключи newfs, указав размер блоков и другие аналогичные параметры. Значения по умолчанию подойдут для большинства случаев:

```
# newfs -v /dev/vinum/test
```

В выводе newfs будет представлен каждый суперблок накопителя.

Создав файловую систему, смонтируйте накопитель и посмотрите, что получится.

```
#mount /dev/vinum/test /mnt
#df
Filesystem       1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad4s1a         248111    55829   172434    24%    /
devfs                    1        1        0   100%    /dev
/dev/ad4s1f        2032839   133492  1736720     7%    /test1
/dev/ad4s1g        2032839  1266476   603736    68%    /test2
/dev/ad4s1h       29497862  3974853 23163181    15%    /usr
/dev/ad4s1e        3048830     6757  2798167     0%    /var
procfs                   4        4        0   100%    /proc
/dev/vinum/test    7282594        1  6699986     0%    /mnt
#
```

В последней строке предыдущего вывода видно, что том Vinum готов к работе, а его объем примерно равен суммарной емкости двух дисков, задействованных для создания тома. (Если вам нужен более удобочитаемый вывод, запустите df -h. Дисковое пространство будет представлено более дружественным способом.)

```
/dev/vinum/test    7282594        1  6699986     0%    /mnt
```

Как видите, объединение дисков полезно при комбинировании нескольких маленьких дисков в один большой. Так старые и маленькие диски могут обрести вторую жизнь. Эта схема не обеспечивает надежность, однако она позволяет вторично задействовать устаревшие диски. Другие варианты применения Vinum, например разбивка на блоки, представляют больший интерес. Рассмотрим создание системы с разбивкой на блоки, взяв для примера те же два диска.

Удаление конфигурации Vinum

Процесс конфигурирования Vinum – *кумулятивный*. Если вы не удалите старую конфигурацию, то новая будет просто добавлена к ней. Кумулятивное конфигурирование отлично подходит для добавления дисков в систему RAID или при восстановлении поврежденного тома. Однако такая схема вызывает затруднения, если вы просто хотите начать с нуля. Это разумно – в конце концов, большинство администраторов действующих систем вносят изменения чаще, чем удаляют конфигурацию и начинают с нуля!

Чтобы начать все с самого начала, необходимо удалить существующую конфигурацию из дисковой базы данных. Редактирование конфигурационного файла не имеет смысла, поскольку он не имеет отношения к внутренним операциям Vinum.

Чтобы избавиться от существующей конфигурации Vinum и монтированных разделов, надо сначала демонтировать существующий раздел Vinum. (Нельзя удалять раздел, пока он используется операционной системой.)

Благополучно размонтировав раздел Vinum, можно удалить существующую конфигурацию. В следующем примере мы задействуем интерактивный режим vinum. Для перехода в оболочку vinum наберите `vinum` в командной строке.

```
# vinum
vinum ->
```

Здесь можно вводить команды vinum(8) и напрямую взаимодействовать с системой. Начнем с команды `resetconfig`:

```
vinum -> resetconfig
 WARNING!  This command will completely wipe out your vinum configuration.
 All data will be lost. If you really want to do this, enter the text

 NO FUTURE
 Enter text ->
(ПРЕДУПРЕЖДЕНИЕ! Эта команда полностью уничтожит конфигурацию vinum.
Все данные будут потеряны. Если вы действительного этого хотите, введите текст

NO FUTURE
Введите текст ->)
```

Да, NO FUTURE заглавными буквами. Таким утонченным способом Vinum напоминает, что вы собираетесь полностью удалить все данные с этого раздела, а также текущую конфигурацию. Если вы засомневались, просто на-

жмите Enter для возврата к основному приглашению vinum. Если вы уверены в своем решении, наберите NO FUTURE и нажмите Enter.

```
Enter text -> NO FUTURE
Vinum configuration obliterated (Конфигурация Vinum уничтожена)
vinum ->
```

Нажмите Control-D для выхода из vinum. Теперь у вас нет конфигурации Vinum и все готово для создания новой.

Тома, разбитые на блоки

Следующий файл *vinum.conf* показывает том, разбитый на блоки. (Напомним, что в таком томе данные распределяются по двум дискам.)

```
drive alpha device /dev/da0s1e
drive beta device /dev/da1s1e
volume test
plex org striped 279k
        sd length 3243037k drive alpha
        sd length 3243037k drive beta
```

Этот файл похож на конфигурационный файл объединенного плекса, – есть лишь незначительные изменения. По-прежнему есть накопители alpha и beta, а также том test.

Место ключевого слова plex org занимают слово striped и число. Число в конце строки plex означает размер блока; это объем данных, который будет записан на один накопитель, прежде чем процесс записи перейдет на другой накопитель. Величина 279k (279 Кбайт), показанная в примере, представляетс собой разумное значение по умолчанию.

Примечание

Зачем нужен такой странный размер блока? Дело в том, что Vinum и файловая система взаимодействуют друг с другом. Если указать четное число, то все ваши индексные дескрипторы разместятся на одном накопителе. Более того, на одном накопителе могут оказаться все данные! В таком случае все преимущества разбивки на блоки будут искоренены. Необходимо убедиться в том, что Vinum использует все накопители. Указанный размер блока обеспечивает именно такое распределение данных.

Однако метки поддисков в предыдущем выводе слегка отличаются от меток в более раннем примере. Поскольку мы разбиваем диски на блоки, поддиски должны иметь одинаковый размер. Если один диск больше другого, то «лишнее» пространство большего диска будет бесполезно, поскольку оба диска будут использовать пространство, доступное на меньшем накопителе. Если вы намереваетесь организовать разбивку на блоки, планируйте разделы Vinum соответствующим образом.

После настройки нового *vinum.conf* запустите vinum create. Вывод vinum create выглядит почти так же, как в предыдущем примере, но слово «concat» заменено словом «striped». Напомним о необходимости тщательно прове-

рить результаты и убедиться, что они соответствуют ожидаемым! Далее запустите newfs, смонтируйте новый том и посмотрите, что получилось.

```
# mount /dev/vinum/test /mnt
# df
Filesystem      1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad4s1a        248111    65146   163117    29%    /
devfs                   1        1        0   100%    /dev
/dev/ad4s1f       2032839   133492  1736720     7%    /test1
/dev/ad4s1g       2032839  1266476   603736    68%    /test2
/dev/ad4s1h      29497862  4193767 22944267    15%    /usr
/dev/ad4s1e       3048830   123222  2681702     4%    /var
procfs                  4        4        0   100%    /proc
/dev/vinum/test   6286319        1  5783413     0%    /mnt
#
```

Новый раздел */mnt* меньше объединенного тома, поскольку мы потеряли дополнительное пространство на одном накопителе. Однако доступ к данным будет ускорен, так как нагрузка распределяется по нескольким дискам. Ускорение доступа к данным будет особенно очевидно, когда множественные процессы будут одновременно записывать данные в том.

Зеркалированные тома

Наконец, займемся зеркалированием наших накопителей. В первую очередь надо размонтировать раздел, удалить старую конфигурацию Vinum с помощью `vinum destroy` и создать новый */etc/vinum.conf*.

```
drive alpha device /dev/da0s1e
drive beta device /dev/da1s1e
volume mirrortest setupstate
plex org concat
        sd length 3243037k drive alpha
plex org concat
        sd length 3243037k drive beta
```

Данная конфигурация немного отличается от предыдущих. Накопители те же, но все остальное выглядит немного по-другому. Например, у тома другое имя, *mirrortest*. Ключевое слово `setupstate` предписывает vinum(8) поверить, что существующее содержимое дисков идентично – мы обманываем Vinum. Обманывать свой компьютер нехорошо, так как обычно он за эту ложь ухватывается и впадает в панику. (Это ведет нас прямо к главе 20.) Однако перед тем как задействовать том, мы создадим в нем файловую систему и перезапишем его содержимое, поэтому Vinum никогда не представится возможность нам поверить.[1]

Сейчас у нас есть два плекса объединенного типа, но в каждом из них есть только один поддиск. Поддиски расположены на отдельных дисках. Под-

[1] Это компьютерный эквивалент выписки чека на оплату аренды накануне дня выплаты.

диск первого плекса находится на накопителе alpha, а поддиск второго плекса – на накопителе beta. Напомним, что они одного размера.

Как известно, плекс – это один экземпляр данных. Поскольку в нашем случае два плекса расположены на отдельных дисках, данные в дисках зеркалированы. Итак, конфигурация готова, и посредством команды vinum можно реализовать ее на жестких дисках.

```
#vinum create vinum.conf
2 drives:
D alpha                 State: up       /dev/da0s1e     A: 384/3551 MB (10%)
D beta                  State: up       /dev/da1s1e     A: 619/3786 MB (16%)

1 volumes:
V mirrortest            State: up       Plexes:       2 Size:       3167 MB

2 plexes:
P mirrortest.p0       C State: up       Subdisks:     1 Size:       3167 MB
P mirrortest.p1       C State: up       Subdisks:     1 Size:       3167 MB

2 subdisks:
S mirrortest.p0.s0      State: up       D: alpha        Size:       3167 MB
S mirrortest.p1.s0      State: up       D: beta         Size:       3167 MB
#
```

Судя по наличию двух плексов, создан зеркалированный том. newfs, mount – и все готово к работе!

Запуск Vinum при начальной загрузке

Для запуска Vinum во время начальной загрузки надо отредактировать */etc/rc.conf* и */etc/fstab*. Параметр `rc.conf` особой сложностью не отличается:

```
start_vinum="YES"
```

Файл */etc/fstab* рассматривался в главе 16. Воспользуйтесь примерами, уже представленными в этом файле, и добавьте в таблицу файловых систем примерно такую строку:

```
/dev/vinum/mailvol   /var/mail     ufs   rw   2  2
```

Другие команды Vinum

Программа vinum(8) включает множество простых команд, позволяющих анализировать состояние Vinum.

Проверка состояния

Самая простая операция – спросить Vinum о его состоянии. Команда `vinum ls` покажет каждый существующий плекс, его размер, имена его накопителей и их состояние:

```
# vinum ls
S mirrortest.p0.s0      State: I 43%    D: alpha        Size:       3167 MB
S mirrortest.p1.s0      State: I 41%    D: beta         Size:       3167 MB
#
```

Каждый накопитель должен находиться в состоянии I. Если это не так, обратитесь к vinum(8) и веб-сайту Vinum (*http://www.vinumvm.org*) – надо выяснить, что означает ошибка. Дополнительную информацию можно получить с помощью команды `vinum list -r`. Она выведет подробную информацию о состоянии каждого тома, плекса и поддиска системы.

Проверка конфигурации

Для надежной проверки конфигурации системы Vinum выполните команду `vinum printconfig`. Будут показаны имена накопителей, затем имя тома и далее плексы и поддиски.

```
#vinum printconfig
# Vinum configuration of magpire.blackhelicopters.org, saved at Fri Aug 17 19:24:43
2001
drive alpha device /dev/da0s1e
drive beta device /dev/da1s1e
volume mirrortest
plex name mirrortest.p0 org concat vol mirrortest
plex name mirrortest.p1 org concat vol mirrortest
sd name mirrortest.p0.s0 drive alpha plex mirrortest.p0 len 6486074s driveoffset
265s plexoffset 0s
sd name mirrortest.p1.s0 drive beta plex mirrortest.p1 len 6486074s driveoffset 265s
plexoffset 0s
#
```

Замена поврежденного зеркалированного плекса

Назначение зеркала – избыточность на случай сбоя накопителя. Как же заменить поврежденный накопитель?[1]

Прежде всего, если возможно, убедитесь в наличии хорошей резервной копии. Вы будете напрямую работать с устройством Vinum, поэтому любая ошибка может разрушить данные. Если текущей копии нет, создайте ее, прежде чем как приступить к дальнейшим действиям. В конце концов, остающийся плекс(ы) должен(ны) работать и обслуживать данные!

Затем надо заменить поврежденный жесткий диск. Это может оказаться трудной задачей. Например, необходимо идентифицировать поврежденный накопитель. Команда `vinum list` выдаст информацию о поврежденном накопителе, сообщив номер соответствующего устройства. Затем имя устройства можно сравнить с именами устройств, приведенными в */var/run/dmesg.boot*, и получить SCSI ID, LUN и номер шины. После идентификации поврежденного накопителя выключите компьютер, найдите соответствующий жесткий диск, замените его на диск с теми же настройками и снова загрузите систему.

---

[1] Работая над этой главой, я уловил высокочастотный шум, исходящий из моего компьютера. Исследование показало, что один из двух жестких дисков SCSI нагрелся, причем настолько, что прикоснувшись к нему, я быстро отдернул руку. Температура в комнате составляла 18 градусов по Цельсию, поэтому стало ясно, что пришло время заменить накопитель.

После перезагрузки убедитесь, что имя устройства для нового накопителя такое же, какое было у старого. (Тогда дальнейшее конфигурирование будет проще.)

Теперь спросим систему Vinum о ее состоянии:

```
# vinum list
1 drives:
D alpha                 State: up       /dev/da0s1e     A: 384/3551 MB (10%)
D beta                  State: referenced               A: 0/0 MB

1 volumes:
V mirrortest            State: up       Plexes:       2 Size:       3167 MB

2 plexes:
P mirrortest.p0       C State: up       Subdisks:     1 Size:       3167 MB
P mirrortest.p1       C State: faulty   Subdisks:     1 Size:       3167 MB

2 subdisks:
S mirrortest.p0.s0      State: up       D: alpha        Size:       3167 MB
S mirrortest.p1.s0      State: stale    D: beta         Size:       3167 MB
#
```

Накопитель beta, которому соответствовало устройство */dev/da1s1*, показан как `"referenced"` (справочный). Это означает, что данный накопитель включен в конфигурацию, но не работает должным образом. В силу каскадной цепочки ошибок плекс на этом накопителе обозначен как `"faulty"` (неисправный), поскольку накопитель неработоспособен. Базовый поддиск обозначен как `"stale"` (устаревший), так как его данные не соответствуют зеркалу.

Здесь нет ничего хорошего, однако все это следовало ожидать. Простая замена не помогла: на дисковом разделе, который мы назвали «beta», ничего нет. Однако наличие ошибок в существующем томе Vinum означает, что в данном накопителе есть неполадки и его следует заменить.

Подготовка заменяющего накопителя

Прежде чем задействовать новый накопитель, на нем необходимо создать разделы и назначить метки. Ранее в этой главе такие операции выполнялись с помощью sysinstall(8). Теперь надо поступить так же. Для начала запустите sysinstall и выберите в меню пункт Fdisk. Далее удалите все дисковые разделы и выделите весь диск для FreeBSD. Затем перейдите в редактор меток (Label editor).

Мы уже настраивали */etc/vinum.conf* и нам известно, что размер раздела Vinum составляет 3 243 037 Кбайт. В примере с `vinum ls` есть более удобное измерение, согласно которому емкость диска составляет 3167 Мбайт. На новом диске для раздела Vinum надо оставить не меньшее пространство. (Я рекомендую по возможности выделить несколько дополнительных мегабайт – на всякий случай.) Теперь запишите изменения на диск, словно вы добавляете новый диск к системе. Затем запустите disklabel и измените тип участка вашего раздела Vinum, как будто вы настраиваете Vinum с нуля.

Примечание

Поскольку емкость дисков растет с каждым днем, вполне вероятно, что заменяющий накопитель будет намного больше первоначального и, возможно, даже быстрее. Оставшейся частью пространства этого диска можно распорядиться по своему усмотрению. Когда на диске остается дополнительное пространство, я зачастую перемещаю туда редко используемые данные – скажем, */usr/src* или */usr/ports/distfiles*. Доступ к ним осуществляется нечасто, поэтому такое размещение не наносит значительного ущерба производительности системы Vinum и, ко всему прочему, высвобождает дисковое пространство в другом месте. Переместите в оставшееся пространство данные, для которых не надо обеспечивать надежность и скорость доступа системы и которые при необходимости легко восстановить.

Восстановление плекса

Наконец, все готово к восстановлению тома. Прежде всего надо уведомить Vinum о наличии нового диска. Напомним, что конфигурация Vinum – кумулятивная. Если просто пересоздать новый том, старая информация добавится сама к себе. Это приведет к неразберихе в Vinum, а данные будут потеряны. Например, в нашем *vinum.conf* есть накопители с именами alpha и beta. Vinum уже знает об этих накопителях. Если бы я задействовал тот же *vinum.conf* и запустил `vinum create` второй раз, Vinum посчитал бы, что у него есть два накопителя alpha и два накопителя beta – оба на одном физическом устройстве! Это привело бы нас в замешательство, не говоря о бедной глупой машине. И, конечно, мы не хотим полностью уничтожить конфигурацию и заставить Vinum забыть о накопителях, которые в нем уже были. Нам надо лишь незначительно изменить существующие настройки.

В данном примере мы хотим восстановить накопитель beta. Поскольку нас не заботит текущее содержимое новенького замещающего накопителя beta, мы можем перезаписать эту часть конфигурации Vinum. Для этого создайте конфигурационный файл Vinum, */etc/vinum.conf.recover*. Укажите в нем только один накопитель без дополнительной информации о плексах и поддисках.

```
drive beta device /dev/da1s1e
```

Это имя накопителя и раздел, на котором он расположен. Теперь добавим эту информацию к существующей конфигурации Vinum, запустив `vinum create`:

```
# vinum create /etc/vinum.conf.recover
2 drives:
D alpha                 State: up       /dev/da0s1e     A: 384/3551 MB (10%)
D beta                  State: up       /dev/da1s1e     A: 2/3170 MB (0%)

1 volumes:
V mirrortest            State: up       Plexes:       2 Size:       3167 MB

2 plexes:
P mirrortest.p0       C State: up       Subdisks:     1 Size:       3167 MB
P mirrortest.p1       C State: faulty   Subdisks:     1 Size:       3167 MB
```

```
2 subdisks:
S mirrortest.p0.s0      State: up       D: alpha        Size:       3167 MB
S mirrortest.p1.s0      State: stale    D: beta         Size:       3167 MB
#
```

Обратите внимание, что состояние накопителя beta теперь "up", а не "referenced". Плекс по-прежнему "faulty", однако прогресс налицо.

Теперь у нас есть накопитель и можно предписать Vinum восстановить диск. Программа Vinum довольно умна; она не запускает накопители, с которыми не может общаться. Нужно лишь указать Vinum, чтобы она запустила замещенный плекс, и программа его восстановит:

```
#vinum start mirrortest.p1
Reviving mirrortest.p1.s0 in the background
# vinum[685]: reviving mirrortest.p1.s0
#
```

С помощью vinum list можно следить за выполнением:

```
# vinum list
2 drives:
D alpha                 State: up       /dev/da0s1e     A: 384/3551 MB (10%)
D beta                  State: up       /dev/da1s1e     A: 2/3170 MB (0%)

1 volumes:
V mirrortest            State: up       Plexes:       2 Size:       3167 MB

2 plexes:
P mirrortest.p0       C State: up       Subdisks:     1 Size:       3167 MB
P mirrortest.p1       C State: faulty   Subdisks:     1 Size:       3167 MB

2 subdisks:
S mirrortest.p0.s0      State: up       D: alpha        Size:       3167 MB
S mirrortest.p1.s0      State: R 11%    D: beta         Size:       3167 MB
#
```

Обратите внимание на последнюю строку: mirrortest.p1.s0. Данные на заменяющем диске восстановлены на 11%. Если зеркало находится под большой нагрузкой, данная операция отнимет определенное время. Однако спустя какое-то время содержимое диска будет восстановлено, и вы сможете вновь приступить к работе с полностью зеркалированными дисками.

Здорово, правда?

18

Производительность системы

«Работает медленно». Это одна из самых страшных фраз, которые может услышать системный администратор. Пользователь не знает, почему система работает медленно – он просто это «чувствует». Обычно пользователь не выполняет контрольную задачу, не воспроизводит те или иные шаги. Нет ничего особенно неверного. Два этих слова могут привести к многочасовому исследованию системы, когда администратор пытается понять, что случилось.

Есть фраза еще более страшная, особенно после того, как на устранение проблемы была затрачена масса времени: «Все равно медленно работает». Неопытный системный администратор считает, что ускорить медленные системы просто – надо приобрести быстродействующие аппаратные средства. Как правило, такое решение помогает устранить проблемы с быстродействием, однако оно потребует больших денежных средств и лишь замаскирует истинные неполадки, а имеющиеся аппаратные средства не найдут должного применения.

В состав FreeBSD входит много инструментов, предназначенных для исследования производительности системы. Они предоставляют информацию, необходимую для выявления истинных причин медленной работы. Они сообщат вам, что нужно предпринять для устранения неполадок. Возможно, вам потребуются более быстрые аппаратные средства, а может быть, удастся перераспределить нагрузку в системе и улучшить общую производительность. Первый шаг состоит в том, чтобы понять истинную причину неполадок.

Ресурсы компьютера

Проблемы быстродействия обычно вызваны запуском большего объема задач, чем может обслужить компьютер. Такое утверждение кажется очевидным, но задумайтесь о нем на минуту. Что оно означает?

В компьютере есть четыре основных ресурса: дисковый ввод/вывод,[1] сетевая пропускная способность, память и процессор. Если возможности одного из

---

[1] Технически сетевая пропускная способность – это часть ввода/вывода. Однако она достаточно специфична, чтобы можно было рассматривать ее отдельно.

ресурсов исчерпаны, то остальные ресурсы не удастся задействовать с максимальным эффектом. Например, процессор может запросто ожидать данных от диска или завершения операции пейджинга в памяти. В таком случае более быстрый процессор не увеличит производительность системы.

Простое обновление аппаратных средств устранит проблемы быстродействия, но не так, как вы предполагаете. Если в системе запускается программа, которая полностью заполняет системную память, то покупка новой системы с более быстрым процессором, вероятно, устранит проблему. В конце концов, в новой системе наверняка больше памяти, чем в старой!

Определив узкое место в системе и направив туда свои усилия, вы сможете выжать из существующего оборудования намного больше. В конце концов, зачем покупать новую систему, если память стоимостью в пару сотен долларов устранит неполадки? (Конечно, если ваша цель – установить эту «медленную» систему в качестве своей настольной машины, то это совсем другое дело.)

Пожалуй, вы сможете перепланировать работу; одна типичная причина замедления работы системы – одновременный запуск нескольких больших программ. Например, однажды я запланировал ротацию протоколов крупной базы данных, которая подразумевала перемещение и сжатие файлов, размер которых составлял гигабайты. Такая ротация затевалась одновременно с автоматическими проверками, которые система выполняла ежедневно. Поскольку задание требовало останова основной базы данных и приводило к простою системы, скорость была очень важна. Производительность обоих процессов невыносимо замедлилась. Перенос ротации протоколов на другое время значительно снизил период простоя.

Рассмотрим несколько инструментов FreeBSD, позволяющих исследовать деятельность системы. Вооружившись такой информацией, обсудим, как устранять неполадки с производительностью. Существуют различные инструменты для исследования каждого потенциального узкого места.

Система FreeBSD непрерывно изменяется; в более новых системах могут появиться новые инструменты настройки производительности. Обратитесь к странице руководства tuning(7) вашей системы и поищите новые советы, относящиеся к производительности. Здесь представлены сведения о настройке, полезные в любой системе FreeBSD (и почти в любой системе UNIX).

Примечание

«Аномальный» (abnormal) – слово, которое постоянно встречается в этой главе. Как системному администратору вам положено знать, что для вашей системы есть норма. Это сродни искусству: можно затрудниться описать норму, но увидев, необходимо уметь ее распознать. С помощью представляемых инструментов полезно регулярно проверять систему, когда она работает исправно, тогда в случае замедления можно разобраться, что не в порядке. Также здесь будут рассмотрены инструменты долгосрочного мониторинга, позволяющие отслеживать производительность системы на протяжении месяцев или лет.

Дисковый ввод/вывод

Дисковые операции подробно рассмотрены в главе 16. Как правило, быстродействие диска существенно ограничивает суммарную производительность вычислительной системы. Такая ситуация, называемая «блокированием на диске», наступает, когда программы вынуждены ожидать завершения дисковых операций, чтобы продолжить обработку данных. В этом случае диск блокирует выполнение программы. Единственное решение состоит в том, чтобы установить более быстрый диск или массив RAID либо распределить дисковые операции между двумя дисками.

Как узнать, блокирует ли диск выполнение программы? Об этом рассказывается в разделе «Применение vmstat» далее в этой главе.

Сетевая пропускная способность

Если производительность системы снижается из-за сети, то необходима более широкая полоса пропускания (например, если заполнен канал T1). Если говорить коротко, то задействовать можно только ту полосу пропускания, которая у вас есть. Если система не может полностью задействовать существующий канал, то для увеличения производительности системы можно применить инструменты, рассмотренные в главе 5.

Выявляя неполадки, начните с мониторинга системы и выясните, какую полосу пропускания она занимает. В главе 15 мы говорили о том, как создавать диаграммы, сообщающие о задействовании полосы пропускания на протяжении длительного периода времени. Кроме того, работа сети рассматривалась в главе 5. Запустите `netstat -m`, изучите вывод этой команды и увеличьте значение параметра ядра `NMBCLUSTERS` (см. главу 4). На самом деле это все, что можно сделать.

Другие компоненты системы отличаются большей сложностью.

Процессор и память

Начать исследование медленно работающей системы лучше при помощи инструмента top(1). Он предоставляет общие сведения о состоянии системы, однако эта информация относится только к использованию процессора и памяти, а ввод/вывод и полоса пропускания не затрагиваются.

Применение top

Для чтения экрана top необходимо хорошо понимать работу системы, поэтому мы потратим определенное время на изучение этого материала. Для запуска top достаточно набрать `top`. Для отображения процессов ядра и пользовательских программ применяется `top -S`. Вывод команды будет очень похож на тот, что приведен ниже. Экран будет обновляться с периодичностью в несколько секунд.

```
# top -S
❶ last pid:   436;  ❷ load averages:  0.14,  0.08,  0.07    ❸ up 0+01:06:16
08:12:26
❹ 46 processes:  3 running, 43 sleeping
❺ CPU states:  1.2% user,  0.0% nice,  0.8% system,  0.0% interrupt, 98.1% idle
❻ Mem: 70M Active, 102M Inact, 26M Wired, 6016K Cache, 41M Buf, 107M Free
❼ Swap: 200M Total, 200M Free
❽
  PID USERNAME PRI NICE  SIZE    RES STATE    TIME   WCPU    CPU COMMAND
  287 mwlucas    2   5  2892K  2136K select   0:13  0.10%  0.10% xsysinfo
  378 mwlucas    2   0   101M 64920K RUN      0:08  0.10%  0.10% soffice.bin
  376 mwlucas    2   0 35372K 32736K RUN      0:13  0.05%  0.05% mozilla-bin
  274 mwlucas    2   0 28208K 26304K select   1:01  0.00%  0.00% XFree86
  170 root       2   0   912K   508K select   0:08  0.00%  0.00% moused
  277 mwlucas    2   0  3888K  3116K select   0:03  0.00%  0.00% wmaker
    5 root      18   0     0K     0K syncer   0:00  0.00%  0.00% syncer
  430 mwlucas   28   0  1912K  1160K RUN      0:00  0.00%  0.00% top
  399 mwlucas    2   0  4500K  4000K select   0:00  0.00%  0.00% Eterm
...
```

Довольно плотно, не так ли? top пытается втиснуть как можно больше данных в стандартное окно терминала 80×25 символов. Экран обновляется каждые две секунды; следовательно, top выдает довольно точный, близкий к реальному времени снимок системы. Остановимся на этом выводе и объясним смысл каждой записи.

Значения PID

Каждый процесс в машине UNIX имеет уникальный идентификатор (ID) процесса, или PID. Когда бы ни стартовал процесс, ему назначается PID, значение которого на единицу превышает PID предыдущего процесса. Поле `last pid` – это идентификатор процесса, запущенного в системе последним. В предыдущем примере в этом поле наблюдается значение `436` (❶). Следующий процесс – `437`, затем `438` и т. д. Наблюдение за инкрементированием позволит обнаружить создание процесса с аномальным номером. Надеюсь, вы обратили внимание, как быстро растет этот номер, когда система работает нормально. Очень быстрый рост значения `last pid` означает, что программы запускаются и останавливаются слишком часто. Это может свидетельствовать об аварийном состоянии того или иного демона или попытке пользователя запускать слишком много программ.[1]

Средняя нагрузка

Поле `load average` (средняя нагрузка; ❷) – это не совсем понятное число. Его назначение – предоставить приблизительные данные о нагрузке системы.

[1] Некоторые пользователи действительно пытаются израсходовать ресурсы системы полностью, запуская все подряд. И закладывая при этом так называемую развилочную *бомбу* (*forkbomb*). Такие пользователи подобны малышам со скриптами, однако они не настолько образованны.

Средняя нагрузка равна сумме среднего количества процессов, ожидающих выделения времени процессора, и среднего количества заданий, ожидающих предоставления доступа к диску. Приемлемое значение средней нагрузки зависит от системы. Если оно чрезмерно велико, значит, надо исследовать работу системы. Многие машины класса x486 сталкиваются с трудностями при средней нагрузке, равной 3, а некоторые современные системы быстро работают при средней нагрузке, равной 10.

top выдает три значения средней нагрузки. Первое (`0.14` в нашем примере) – это средняя нагрузка за последнюю минуту. Второе (`0.08`) – за последние 5 минут, а третье (`0.07`) – за предыдущие 15 минут. Если средняя нагрузка за последние 15 минут высока, а за последнюю минуту – низка, то налицо высокий пик активности системы, который пришелся на этот 15-минутный интервал. Как показала себя система? С другой стороны, если 15-минутное значение невелико, а нагрузка за последнюю минуту высока, значит, что-то произошло за последние 60 секунд и, возможно, продолжается сейчас. Если велики все значения средней нагрузки, то такое состояние удерживалось все 15 минут.

uptime

Последнее поле в первой строке – *uptime* (❸). Оно сообщает о том, какой период времени работает система. В нашем примере система работает один час и шесть минут, а текущее время – 08:12:26. Предоставляю вам возможность вычислить, когда произошла загрузка системы.

Количество процессов

Во второй строке представлена информация о процессах, запущенных в системе в настоящее время (❹). Запущенные (running) процессы непосредственно выполняют работу; они отвечают на запросы пользователей, обрабатывают почту и выполняют все другие операции. Спящие (sleeping) процессы ожидают входных данных от того или иного источника. Это хорошие состояния процессов. Процессы в других состояниях обычно ожидают доступности ресурсов или в некотором роде зависли. Большое количество неспящих, незапущенных процессов может свидетельствовать о неполадках. Продолжите исследование и определите, что это за процессы.

Типы процессов

Строка `CPU states` (❺) сообщает, какая доля доступного времени процессора (в процентах) расходуется на обслуживание процессов различных типов и другие задачи. В этой строке пердставлены пять различных типов процессов: `user`, `nice`, `system`, `interrupt` и `idle`.

Пользовательские процессы (`user`) – это средние повседневные программы. Это могут быть демоны, запущенные от имени root, команды, запущенные обычными пользователями, или что-нибудь другое. Если процесс представлен в списке системных процессов (то есть в выводе `ps -ax`), это пользовательский процесс.

Процессы nice (nice) – это процессы, приоритет которых определяется пользователем. Более подробно они рассмотрены в разделе «Изменение приоритетов с помощью nice».

Системные процессы (system) – это процессы ядра. В состав таких процессов входят обработчики виртуальной памяти, сетевые процессы, процессы записи на диск и т. д.

Категория прерываний (interrupt) показывает, сколько времени система затрачивает на обработку запросов прерывания (IRQ).

Наконец, поле *бездействующих процессов* (idle) показывает, сколько времени система ничего не делает. Если значение idle процессора очень мало, то стоит задуматься о перепланировании заданий или покупке более быстрого процессора.

Примечание

При работе с многопроцессорной системой следует иметь в виду, что top отображает средние показатели нагрузки по всем процессорам. Один процессор может быть полностью занят компиляцией, а другой может бездействовать. В этом случае top покажет, что система загружена на 50%.

Память

Далее идет строка Mem, представляющая действительную физическую память (❻). В отличие от Windows, где память просто разделена на категории «используемая» (used) и «неиспользуемая» (unused), FreeBSD применяет память разными способами.

Активная память (Active) – это общая емкость памяти, которая в данный момент задействована для выполнения пользовательских программ и хранения их данных. Когда выполнение програмы завершено, ее данные попадают в *неактивную* память (Inact). Данные, полученные с диска, помещаются в *кэш* (Cache).

Аналогично, запись Buf показывает емкость буфера памяти. Буфер памяти содержит данные, недавно полученные с диска.

Свободная память (Free) не задействована.

Связанная память (Wired) – это память, применяемая для структур данных ядра, а также для особых системных вызовов, которым память нужна немедленно. Связанная память никогда не свопируется.

Своп

Далее идет строка Swap (❼), содержащая данные о доступном пространстве свопа и о задействованном объеме. При *свопинге* (*swapping*) система использует дисковый накопитель как дополнительную память. Далее в этой главе своп будет рассмотрен более подробно.

Список процессов

Наконец, top выдает список процессов системы и их базовые характеристики (❸). Формат таблицы подразумевает выдачу как можно большего объема информации в максимально сжатой форме. Каждый процесс описан в отдельной строке. Рассмотрим каждую колонку.

`PID`

Первая колонка – номер ID, или `PID`. Каждый процесс, запущенный в системе, имеет уникальный PID. Выполняя команды kill, вы задаете PID для идентификации процесса, на который хотите воздействовать.

`USERNAME`

Следующая колонка – имя пользователя, от лица которого запущен процесс. Если несколько процессов с одним пользовательским ID интенсивно потребляют время процессора или память, вы знаете, с кем поговорить.

`PRI` и `NICE`

Колонки `PRI` (priority, приоритет) и `NICE` взаимосвязаны. Они показывают, какой приоритет система назначает тому или иному процессу. О приоритетах и nice речь пойдет чуть далее.

`SIZE`

Это объем памяти, который система выделила для этого процесса.

`RES`

В колонке `RES` (`Resident Memory`) показано, какая часть программы действительно находится в памяти (резидентна). Для программы может быть отведен огромный объем памяти, но задействована ею будет лишь малая его часть.

`STATE`

В колонке `STATE` показано, что делает процесс в данный момент. Процессы могут пребывать в разных состояниях: ожидать ввода данных; спать, пока их кто-нибудь не разбудит; выполняться и т. д.

`TIME`

Колонка `TIME` сообщает, в течение какого периода времени выполняется процесс.

`WCPU` (Использование CPU)

Колонка `WCPU` выдает взвешенные показатели использования CPU, а именно процентное отношение времени CPU, которое процесс использует согласно своему приоритету и значению nice. Здесь показано, какой процент времени CPU программа действительно использует.

`COMMAND` (Имя команды)

Наконец, в колонке `COMMAND` представлено имя программы.

Использование памяти

Если система работает медленно, прежде всего проверьте показатели использования памяти и процессора. Эти компоненты не более склонны выходить из-под контроля, чем другие, но их легче всего измерить. Сначала рассмотрим память.

FreeBSD ошибается при кэшировании данных (что ведет к замедлению работы системы), к которым недавно осуществлялся доступ, если с диска постоянно считывается неожиданно большой объем информации. Если эту информацию можно кэшировать в физической памяти, то доступ к ней осуществляется очень быстро. Если системе требуется больше памяти, она выгружает из кэша наиболее старые записи, освобождая место для новых данных.

Например, обсуждаемый вывод top взят из моего лэптопа, который использует много буферов и неактивной памяти. Частично это объясняется работой веб-броузера. Я запустил Mozilla утром предыдущего дня при загрузке системы, чтобы посмотреть утренние комиксы.[1] Недолгое время индикатор обращения к диску непрерывно мигал, пока программа считывалась с диска. Затем я остановил броузер, чтобы приступить к работе.

Поскольку к веб-броузеру был произведен доступ, он оказался в буферном кэше. Когда сегодня утром я запустил его снова, он был вызван из кэша, а не с диска, поэтому его запуск прошел намного быстрее. Запусти я другой большой процесс, он «вытолкнул» бы веб-броузер из кэша, чтобы считать больше данных.

Если ваша система работает как полагается, у вас есть как минимум несколько мегабайт свободной памяти. Если свободной памяти больше, значит, ваша система не в полной мере задействует свой потенциал. В предыдущем примере я избавился от 128 Мбайт оперативной памяти, и это никак не повлияло на производительность системы.

Если в системе достаточно места в кэше или буфере, то она не испытывает нехватки памяти. Дополнительная память может принести пользу, но ее наличие не строго обязательно. Если активная и связанная память составляют большую часть всей доступной, то дополнительная оперативная память не помешает.

В случае нехватки свободного пространства и минимума места (или его отсутствия) в кэше или буфере, необходимо выяснить, на что расходуется память. Вполне вероятно, что системе ее не хватает. Обратитесь к разделу «Применение vmstat» далее в этой главе.

Потребление пространства свопа

Виртуальная память, или *своп* (*swap*), помогает справиться с кратковременными нехватками памяти. Например, разархивируемый огромный файл может

---

[1] Sluggy Freelance (*www.sluggy.com*) и Help Desk (*www.ubersoft.net*), если это кому-то интересно.

легко заполнить всю физическую память, и системе придется задействовать виртуальную. Нет необходимости в покупке дополнительной памяти, если использоваться она будет лишь время от времени, а можно обойтись свопом.

Как и кэш памяти, своп кэширует данные, которые обрабатывались недавно. И если к свопу уже было выполнено обращение, он не освободится никогда. Например, один сервер на момент написания этих строк проработал 772 дня. Однажды я задействовал около сотни мегабайт свопа для обслуживания большого процесса компиляции. Экран top по-прежнему показывает, что используется 100 Мбайт свопа, хотя в системе свободно свыше 200 Мбайт памяти.

Применять свопинг не так уж плохо, особенно если программа 80% времени тратит на выполнение 20% кода. Поскольку значительная часть оставшегося времени тратится на запуск программы, ее останов и обработку ошибок, в этих операциях можно благополучно задействовать пространство свопа. Влияние такого шага на выполнение программы будет минимальным.

Таким образом, совершенно нормально, если время от времени система обращается к свопу. Однако если система постоянно прибегает к свопу, возможно, надо установить больше памяти.

Использование процессора

Процессор может выполнять ограниченное количество операций в секунду. Если требуется выполнить больше операций, чем может обслужить процессор, запросы выстраиваются в очередь. Они будут накапливаться, а система замедлит свою работу. Так, в двух словах, используется процессор.

Если top показывает, что процессор занят около 100% своего времени, надо предпринять определенные шаги. Один из вариантов состоит в покупке новых аппаратных средств, но есть и другие. Например, можно изучить процессы, запущенные в системе, и выяснить, все ли они необходимы. Не установил ли младший администратор клиент SETI@Home (*/usr/ports/astro/setiathome*), чтобы искать инопланетян, затрачивая циклы процессора? Запущены ли программы, ранее имевшие важное значение, но не утратившие его? Найдите и убейте (kill) ненужные процессы. Кроме того, убедитесь, что они не будут запущены при следующей загрузке системы.

Выполнив эти шаги, снова измерьте производительность системы. Если неполадки по-прежнему есть, попробуйте перепланировать процессы или изменить их приоритеты.

Перепланирование

Перепланирование легче изменения приоритетов. Это относительно простой способ сбалансировать процессы системы, чтобы они не перегружали процессор. Мы уже говорили в главе 9, что для планирования запуска системных заданий в различное время можно задействовать cron(1), однако пользователи тоже могут планировать процессы. Если тот или иной пользователь запускает большие задания на компиляцию или делает гигантские запросы к базе данных, то с помощью cron выполнение этих заданий можно перене-

сти на ночь. Зачастую такие задания, как ежемесячный поиск по базе данных счетов, можно запускать между 6 часами вечера и 6 часами утра, и это никому не помешает. Аналогично запуск `make buildworld && make buildkernel` можно назначить на час ночи.

Изменение приоритетов с помощью nice

Если перепланирование не помогло, остается изменение приоритетов, отличающееся немного большей сложностью. Изменяя приоритет, вы предписываете UNIX изменить значимость данного процесса.

Пусть требуется запустить установку программы лишь в том случае, если нет более важных заданий. В этом случае надо изменить приоритет процесса установки с помощью «любезности» (niceness), которая представляет собой относительную оценку процессорного времени, необходимого процессу. Чем любезнее процесс, тем меньше процессорного времени он запрашивает. По умолчанию любезность равна 0, но ее значение может находиться в диапазоне от 20 (очень любезный процесс) до –20 (совсем нелюбезный). (Такая система может показаться поставленной с ног на голову – ведь можно возразить, что больший номер должен означать больший приоритет. Это, однако, порождает и языковую проблему, так как предложение назвать этот показатель «неуживчивостью» или «жадностью» не показалось удачным, когда название придумывали.)[1]

На экране top, представленном ранее (в разделе «Применение top»), приоритеты процессов показаны в колонке `PRI`. FreeBSD вычисляет приоритеты процессов, учитывая различные факторы, и в том числе любезность. При любой возможности система старается запускать первыми процессы с высокими приоритетами. Любезность оказывает влияние на приоритеты, однако приоритеты нельзя изменять напрямую.

Если известно, что система работает медленно из-за нехватки мощности процессора, запускаемым программам можно назначить приоритет с помощью nice(1). Укажите желаемый уровень любезности, поставив одиночное тире перед командой. Например, для запуска `make buildworld` с уровнем любезности 15 выполните такую команду:

```
# cd /usr/src
# nice -15 make buildworld
```

Отрицательные значения любезности может назначать только root. Для запуска программы с отрицательным значением любезности укажите двойное тире (`nice --5`). Например, если требуется как можно скорее скомпилировать и наложить на ядро важную заплатку, укажите отрицательное значение любезности:

```
# cd /sys/i386/compile/MYKERNEL
# nice --20 make depend && nice --20 make all install
```

[1] Возможно, это один из немногих случаев, когда при выборе имени команды UNIX возобладал здравый смысл.

Обычно при запуске программы не надо указывать уровень любезности, однако нередко возникает потребность изменить уровень любезности на лету (как правило, когда обнаруживается, что процесс полностью оккупировал CPU). Это можно сделать командой renice(8), позволяющей изменять приоритет с помощью указания PID или владельца процесса. Для изменения уровня любезности процесса надо запустить renice, указав новое значение любезности и идентификатор процесса.

Например, одна из моих систем поддерживает зеркало FreeBSD CVSup. Если я нахожу, что зеркало потребляет слишком много процессорного времени, я могу назначить этому заданию уровень любезности 20 (это максимально возможное значение). По сути, это означает, что данную команду следует запускать, лишь если никаким другим программам процессорное время не требуется. Для изменения приоритета запущенного процесса надо знать его ID. Я знаю, что процесс называется cvsupd, поскольку я запускал top на протяжении последних нескольких месяцев. Затем изучаю список всех процессов, запущенных в системе, и нахожу процесс cvsupd с помощью следующей команды:

```
# ps -ax | grep cvsupd
  322  ??  Is     0:00.01 /usr/local/sbin/cvsupd -C 5 -b /test2 -s sup
#
```

Значение в первой колонке предыдущего вывода `ps` — это идентификатор процесса, `PID 322`. Для изменения уровня его любезности я выполняю такую команду:

```
# renice 20 322
322: old priority 0, new priority 20
#
```

Бум! Демон cvsupd теперь будет работать, если никакой другой запущенной программе не требуется системное время. Конечно, это очень раздосадует пользователей сервиса, однако для такого решения у меня есть свои причины. (Поскольку это частное зеркало, а не общедоступное, я не обязан быть особенно любезен со своими пользователями.)

Чтобы изменить уровень любезности всех процессов, запущенных тем или иным пользователем, задействуйте ключ `-u`. Например, чтобы придать моим процессам большую значимость по сравнению со всеми другими процессами, я могу ввести такую команду:

```
# renice -5 -u mwlucas
1000: old priority 0, new priority -5
#
```

Число `1000` — это мой пользовательский ID в этой системе. Снова отмечу, что у меня есть достаточные основания так поступить. И дело здесь не только в стремлении проявить собственное могущество.[1]

---

[1] Эгоизм нельзя считать достаточным основанием. По крайней мере, мне так говорили.

Примечание

Изменение уровня любезности, перепланирование и управление процессами не приводят к появлению дополнительного времени процессора, а переназначают его. Если не удается перепланировать процессы и изменить уровни любезности должным образом, необходимо заменить аппаратные средства на более быстрые или добавить новые. В некоторых системных платах есть гнездо для дополнительного процессора. Это быстрый и недорогой способ увеличить производительность, когда системе требуется много процессорного времени. Обладателям систем с несколькими процессорами необходимо почитать об SMP в главе 11.

Когда своп некстати

Ранее я упоминал, что в применении свопинга нет ничего плохого. Пространство свопа используется как виртуальная память. (Другими словами, пространство на жестком диске используется так же, как и оперативная память.) Жесткий диск намного медленнее оперативной памяти, однако своп, так или иначе, работает, а многим программам вовсе не надо хранить все данные в оперативной памяти. Если 80% своего времени программа тратит на 20% своего кода, то 80% кода можно разместить в свопе без серьезного снижения производительности.

Многие системные администраторы применяют термин «свопинг» в общем смысле, смешивая две разные операции (пейджинг и свопинг) и не понимая разницы между ними.

Пейджинг

В книгах, рассказывающих о виртуальной памяти, упоминаются *страницы* (*pages*). Страница – это просто участок памяти. В случае FreeBSD на платформе x86 размер страницы равен 4 Кбайт. (Различные платформы имеют различные размеры страниц.)

Между реальной и виртуальной памятью данные перемещаются постранично. *Пейджинг* (*paging*) подразумевает перенос части работающей программы в своп. Такой процесс может улучшить производительность в тяжело нагруженной системе, поскольку незадействованные данные могут храниться на диске, пока в них не возникнет необходимость.

Свопинг

Свопинг (*swapping*) подразумевает перенос в своп всего запущенного процесса. Если в компьютере недостаточно физической памяти для хранения процесса, который не работает в данную микросекунду, система может переместить весь процесс в своп. Затем, когда процессор запустит этот процесс в следующий раз, образ процесса будет перенесен из свопа в физическую память, а в своп будут отправлены другие процессы.

Сложности со свопингом заключаются в том, что использование диска поставлено с ног на голову, а производительность значительно падает.

Поскольку запросы обрабатываются дольше, в системе появляется больше процессов. Протоколирование с целью изучения проблемы лишь усугубляет ситуацию, поскольку протоколирование – это дополнительный системный процесс. Такое влияние на производительность иногда нызывают *спиралью смерти* (*death spiral*).

Нехватка памяти снижает производительность системы больше, чем любой другой фактор. Если свопинг происходит часто, вы *должны* приобрести дополнительную память либо смириться с непотребной[1] производительностью.

Примечание

В каждой системе есть узкие места, ограничивающие производительность. Если устранить одно узкое место, производительность будет увеличиваться, пока не настанет очередь следующего узкого места. Производительность системы ограничена скоростью самого медленного компонента («привязана» к его скорости). Так, нередко скорость веб-сервера определяется пропускной способностью сети, поскольку самый медленный компонент системы – это соединение с Интернетом. Если установить более быстрое соединение с Интернетом, то система сможет выдавать веб-страницы с такой скоростью, с какой ей это позволят процессор и диск.

Что это: свопинг или пейджинг?

В состав FreeBSD входят несколько программ для исследования производительности системы. Среди них есть vmstat(8), iostat(8) и systat(1). Здесь будет рассматриваться vmstat, поскольку, по моему мнению, этот инструмент наиболее полезен. iostat подобен vmstat, а systat предоставляет сходную информацию в более удобном формате.

Применение vmstat

vmstat(8) предоставляет статистику использования виртуальной памяти в текущий момент. К выводу vmstat надо привыкнуть, но он очень хорош тем, что предоставляет большой объем данных в маленьком пространстве. Наберите `vmstat` в командной строке и посмотрите, что получилось.

```
# vmstat
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad4 da0   in   sy  cs us sy id
 0 0 0    7096 479140   21   0   0   0   9   0   0   0  331  102 437  0  1 99
#
```

Экран разбит на шесть разделов: процесс (`procs`), память (`memory`), пейджинг (`page`), диски (`disks`), ошибки (`faults`) и процессор (`cpu`). Кратко рассмотрим каждый раздел, а затем углубимся в вопросы, имеющие наибольшеее значение при исследовании производительности.

[1] Я бы употребил слово получше, чем «непотребный», но мой редактор не одобряет язык «с изюминкой», которому я научился у коллеги, служившего на флоте.

Процессы

Под заголовком `procs` есть три колонки.

`r`

Количество процессов, ожидающих времени процессора. Они готовы к запуску, но не могут получить доступ к процессору. Если их много, то узким местом системы является процессор.

`b`

Количество процессов, блокированных в ожидании системного ввода или вывода. Обычно это ожидание доступа к диску. Такие процессы будут запущены сразу после получения необходимых данных. Если это число велико, узким местом является диск.

`w`

Показывает процесы, которые запущены, но полностью свопированы. Если процессы начинают свопироваться регулярно, размер памяти не соответствует заданиям, выполняемым в системе.

Память

В разделе `memory` есть две колонки.

`avm`

Среднее количество страниц виртуальной памяти, используемых системой. Если это значение чрезмерно велико, значит, система расходует виртуальную память.

`fre`

Количество страниц, доступных для использования. Если это значение чрезмерно мало, налицо проблемы с памятью.

Пейджинг

Раздел `paging` показывает, как работает система виртуальной памяти.

`flt`

Количество ошибок из-за отсутствия страницы (page faults), когда необходимой информации нет в памяти и приходится загружать ее с диска.

`re`

Количество страниц, восстановленных из кэша и повторно использованных.

`pi`

Сокращение от *page in*. Показывает, сколько страниц было перемещено из физической памяти в своп.

`po`

Сокращение от *page out*. Показывает, сколько страниц было перемещено из свопа в реальную память.

`fr` и `sr`

Показывают, сколько страниц высвобождено и просмотрено в секунду соответственно. На эти значения надо обращать внимание лишь тогда, когда система находится под очень большой нагрузкой.

Диски

Раздел `disks` показывает все диски по именам устройств. Здесь представлено количество дисковых операций в секунду. При любой возможности необходимо распределять дисковые операции между различными дисками, а диски размещать на разных шинах (см. главы 1 и 13). Если один диск загружен больше остальных, а система ожидает доступа к диску, то наиболее часто используемые файлы надо переместить с одного диска на другой.

Ошибки

В разделе `faults` представлены системные ошибки. В данном случае в ошибках нет ничего плохого, они представляют собой всего лишь полученные системные прерывания.

`in`

Количество системных прерываний (запросов IRQ), полученных системой за последние пять секунд.

`sy`

Количество системных вызовов за последние пять секунд.

`cs`

Количество *переключений контекста* (*context switches*). Другими словами, количество переходов процессора от выполнения одного задания к выполнению другого.

Процессор

Наконец, в разделе `cpu` показано, сколько времени система потратила на выполнение пользовательских (`us`) и системных задач (`sy`) и сколько времени система бездействовала (`id`). Это та же информация, которую представляет top.

Как извлечь пользу из информации vmstat

Как же применить эту информацию? Прежде всего проверьте первые три колонки и выясните, чего ждет система, когда работает медленно. Если она ожидает доступа к процессору (колонка `r`), значит, не хватает мощности процессора. Если система ждет завершения дисковых операций (колонка `b`), то узким местом являются диски. Если часто происходит свопинг (колонка `w`), то в системе не хватает памяти. Довольно просто, не правда ли?

Если в системе маловато памяти, то можно предположить, что в разделе `page` будут очень высокие значения. (Подробности управления виртуальной памятью засекречены и не будут здесь рассматриваться в деталях.) Необходимо знать главное – что для системы есть норма и что для нее не есть норма.

Мониторинг нескольких дисков

vmstat показывает, что происходит с дисками и куда записываются данные. Количество дисковых операций в секунду – это важный параметр, показывающий, насколько хорошо диски справляются с возложенной нагрузкой.

Однако при наличии нескольких дисков можно заметить, что они не появляются в выводе vmstat, которая старается уместить свой вывод в 80 колонках и поэтому не может представить все диски системы. Но при помощи ключа -s можно выйти за эти пределы и задать желаемое количество накопителей. Предел в 80 колонок важен на системной консоли, однако его можно легко преодолеть, воспользовавшись SSH с рабочей станции.

Непрерывное выполнение vmstat

Применяя vmstat, вы наверняка больше заинтересованы в получении информации за определенный период времени, нежели в изучении состояния системы в тот или иной момент. С помощью ключа -w программу vmstat можно запустить как непрерывно обновляющийся экран, указав количество секунд между обновлениями. Значения пересчитываются многочисленными внутренними счетчиками системы каждые пять секунд, поэтому именно пять секунд – это минимальное рекомендуемое время между обновлениями.

```
# vmstat -w 5
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad0 md0   in   sy  cs us sy id
 1 0 0  165208  51408  431   0   0   0 408   4   0   0  243 2656 255 13  3 83
 0 0 0  165208  51408    8   0   0   0   0   0   0   0  267  829 232  0  2 97
 1 0 0  172480  51408    9   0   0   0   2   0   0   0  277  986 279  2  1 97
 1 0 0  174584  51108   44   0   0   0  21   0   0   0  262 3694 269  1  3 96
...
```

Нажмите Control-C, когда будете готовы, и просто сидите и наблюдайте работу системы. Обратите внимание на реакцию системы, когда начнут выполняться запланированные задания. В этом примере представлена редкая ситуация, когда процессы ожидают времени процессора (что показывают единицы в колонке 1), однако диск и память якобы работают хорошо. Если процессы изредка ожидают освобождения того или иного ресурса, то это еще не означает, что данный системный компонент нуждается в обновлении. Не обращайте на это внимания, если производительность системы приемлема.

Настройка производительности на практике

Все эти теоретические выкладки замечательны, но как настраивать производительность в реальной жизни? Сейчас я надеюсь, что вы действительно прочли предыдущие главы книги, поскольку далее я буду ссылаться на информацию, представленную в различных местах. Выполним реальный тест на реальной системе, чтобы продемонстрировать настройку производительности.

Стандартный тест FreeBSD с высокой нагрузкой – это процесс `make world`, запускаемый при выполнении обновления из исходного кода. Он нагружает процессор и диск, а также поглощает всю память, которую ему удастся занять. Сосредоточим внимание на этапе `make buildkernel` этого процесса, поскольку он короче этапа `make buildworld`, который лучше подходит для тестирования. Рассмотрим, как задействовать методы и информацию, представленные здесь, для снижения времени выполнения сборки.

Система, на которой я буду выполнять тест, имеет два процессора Pentium, работающих на частоте 1 ГГц, новый диск IDE емкостью 60 Гбайт, два почтенных диска SCSI-2 емкостью 4,5 Гбайт и 542 Мбайт оперативной памяти. В системе запущены несколько маленьких программ, но нет сервера X, менеджера окон и веб-сервера. Первоначально система установлена на одном диске IDE; диски SCSI полностью бездействуют. В основном в системе оставлены значения параметров по умолчанию, а в резделе */usr* установлены Soft Updates (см. главу 16).

Честность при тестировании

Тестирование – трудная задача. Мы не будем проводить формальное тестирование, однако обеспечим условия, при которых результаты каждого запуска нашего теста будут по возможности объективными. Между тестами я буду перезагружать систему, чтобы удалить все, что может скрываться в буферном кэше. В данном случае мы хотим увеличить чистую производительность, не задействуя буферизацию и кэширование FreeBSD. В интервалах между тестами я буду удалять каталог */usr/obj* (где buildkernel создает свои файлы).

Примечание

Хотя `make buildkernel` – это достаточно стандартный тест, не считайте его самым важным показателем производительности FreeBSD. Здесь он рассматривается потому, что это стандартный процесс, к которому есть доступ у каждого. Тестируйте свои системы с помощью программ и команд, которые вы действительно используете, а не с помощью произвольных оценок.

Первоначальный тест

Прежде всего будем записывать временные срезы начала и завершения каждого запуска. Они дадут нам абсолютную оценку влияния тех или иных изменений на производительность:

```
# date >> timestamps && make buildkernel && date >> timestamps
```

Эта последовательность команд записывает время начала и завершения теста в файл *timestamps* и запускает `make buildkernel`.

Сейчас запустим сборку и обратим внимание на вывод top. Здесь представлены первые несколько строк:

```
last pid:  6262;  load averages:  0.87,  0.37,  0.15    up 0+01:00:43  12:14:17
46 processes:  2 running, 44 sleeping
```

```
CPU states: 21.2% user,  0.0% nice, 29.4% system,  0.6% interrupt, ❶ 48.8% idle
Mem: 16M Active, 38M Inact, 36M Wired, 2240K Cache, 61M Buf, ❷ 407M Free
Swap: 2048M Total, 2048M Free

  PID USERNAME PRI NICE   SIZE    RES STATE  C   TIME   WCPU    CPU COMMAND
  529 root      96    0  2420K  1956K ect    0   0:01  0.10%  0.10% sshd
  275 root      96    0  1020K   588K ect    0   0:08  0.00%  0.00% moused
  354 root      96    0  2420K  1956K ect    1   0:00  0.00%  0.00% sshd
  223 root       8    0  1084K   688K slp    0   0:00  0.00%  0.00% diskcheckd
  252 root      96    0  2364K  1676K ect    1   0:00  0.00%  0.00% sshd
...
```

Можно сразу заметить, что в системе нет нехватки памяти; 407 Мбайт свободного пространства (❶) будет достаточно на некоторое время. Процессор бездействует на 48,8% (❷); следовательно, время процессора не является узким местом.

Рассмотрим фрагмент вывода vmstat, обновляемый каждые пять секунд:

```
# vmstat 5
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad4 da0   in   sy  cs us sy id
 0 2 0   17952 396524  173   0   0   0 199   0   0   0  354  385 630  2  2 96
 2 1 0   18872 394908 2260   0   0   0 2281  0  31   0  369 2682 2342 37 16 47
 2 2 0   19268 394384 1801   0   0   0 1856  0   2   0  336 2107 1687 40 12 48
 0 2 0   19164 393768 2074   0   0   0 2143  0  15   0  353 2617 2162 32 14 53
 2 2 0   21680 389032 2045   0   0   0 1892  0   1   0  337 2349 1908 40 12 47
 0 2 0   16096 393452 1916   0   0   0 2242  0   1   0  338 2281 2240 39 14 47
 2 2 0   17888 390616 2260   0   0   0 2236  0   1   0  342 2830 2844 35 17 47
 1 2 0   18880 389728 2260   0   0   0 2337  0  30   0  370 2804 2909 35 19 46
 0 2 0   16484 391684 2031   0   0   0 2234  0   1   0  338 2477 2183 37 16 47
 1 2 0   18416 389052 2230   0   0   0 2219  0  11   0  352 2886 2876 33 18 49
...
```

Да, что-то здесь не так. Колонка `r` показывает, сколько запущенных процессов не могут получить времени процессора. Наша система бездействует почти на 50%, однако некоторые процессы не могут получить время процессора! В чем дело?

Так, это многопроцессорная система. Напомним, что процессор за раз выполняет только одну операцию. Здесь мы видим, что один процессор действительно загружен, а другой вообще ничего не делает. Решение – распределить нагрузку между нашими процессорами. Как говорилось в главе 6, для make можно применить ключ `-j`. Так и поступим в следующем тесте.

Когда выполнение `make buildkernel` завершится, обратите внимание на временные срезы нашего теста:

```
#more timestamps
Sun Aug 19 12:11:47 EDT 2001
Sun Aug 19 12:23:43 EDT 2001
#
```

Всего 12 минут без 4 секунд, или 716 секунд. Вот так, но можем ли мы улучшить показатели?

Применение двух процессоров

Попробуем задействовать *оба* процессора и посмотрим, что произойдет со временем.

```
# date >> timestamps && make -j2 buildkernel && date >> timestamps
```

Для ключа `j` можно задать значения больше 2, однако начать лучше именно с 2. В теории такой ключ должен извлечь намного больше из наших процессоров.

Посмотрим, как теория воплощается на практике:

```
last pid:  3855;  load averages:  1.08,  0.36,  0.16    up 0+00:07:18 12:36:45
51 processes:  1 running, 47 sleeping, 1 zombie, 2 mutex
CPU states: 23.0% user,  0.0% nice, 41.2% system,  0.0% interrupt, ❶ 35.8% idle
Mem: 16M Active, 14M Inact, 29M Wired, 8K Cache, 51M Buf, 442M Free
Swap: 2048M Total, 2048M Free

  PID USERNAME PRI NICE   SIZE    RES STATE  C   TIME   WCPU    CPU COMMAND
 2016 root       8    0   612K   512K t      1   0:01  1.46%  1.22% cc
  282 root      96    0  1020K   588K ect    1   0:01  0.00%  0.00% moused
 1653 root      96    0  1756K  1632K ect    1   0:01  0.00%  0.00% make
  351 root      96    0  2420K  1952K ect    1   0:01  0.00%  0.00% sshd
...
```

Бездействие на 35,8% – намного лучший результат (❶) (теперь работают оба процессора). Запустим vmstat:

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad4 da0   in   sy  cs us sy id
 0 4 0   19236 385696 2521   0   0   0 3021   0   8   0  350 3691 5099 52 27 21
 0 5 0   29304 374976 2768   0   0   0 2381   0  14   0  357 3929 5403 56 28 16
 0 2 0   20992 381444 3140   0   0   0 3634   0  25   0  364 4143 6585 52 32 16
 0 1 0   19348 378600 2925   0   0   0 3064   0  39   0  386 4542 8170 39 34 27
 0 6 0   25296 372936 3464   0   0   0 3356   0   7   0  349 4734 9110 47 37 16
 0 2 0   20456 374884 2584   0   0   0 2860   0  44   0  383 3683 5209 52 27 21
 0 6 0   27140 367312 2828   0   0   0 2660   0  10   0  352 4081 6572 52 32 16
...
```

Да, с процессорами дело пошло получше. Если наблюдать достаточно долго, можно заметить редкие случаи нехватки процессорного времени, но ситуация намного лучше предыдущей. (Непродолжительная нехватка времени совершенно естественна, и надо обращать внимание лишь на постоянную нехватку.) Однако вызывает тревогу содержимое колонки `b`. В ней представлены процессы, которые не могут выполняться, поскольку ожидают доступа к диску. Значение в колонке ни разу не оказалось равным 0. Следовательно, процесс попал в зависимость от диска.

Тем не менее рассмотрим текущие временные срезы:

```
Sun Aug 19 12:35:25 EDT 2001
Sun Aug 19 12:46:46 EDT 2001
```

Всего 11 минут 21 секунда, или 681 секунда. Использование обоих процессоров позволило сократить продолжительность процесса примерно на 30 секунд. Хотя это эквивалентно лишь 5-процентному увеличению быстродействия, мы добились результата без трат на покупку дополнительного оборудования. Следующая задача – избавиться от узкого места, создаваемого вводом/выводом.

Кэширование каталогов

Наименее обременительный способ снижения дискового ввода/вывода состоит в том, чтобы включить кэширование каталогов с помощью sysctl `vfs.vmiodirenable` (см. главу 16). С помощью sysctl(8) это можно сделать без перезагрузки системы (см. главу 4). (Возможно, данная функция уже включена, но это надо проверить. Все зависит от версии FreeBSD.)

```
# sysctl vfs.vmiodirenable=1
vfs.vmiodirenable: 0 -> 1
#
```

Поскольку, согласно условиям тестирования, нам придется перезагрузить систему, установим этот параметр sysctl в */etc/sysctl.conf*, удалим */usr/obj* и перезагрузим систему. Затем запустим `make buildworld` с помощью той же команды. Вывод top почти идентичен выводу при последнем запуске, поэтому здесь он не приводится. Вывод vmstat также подобен предыдущему случаю. Вот временные срезы:

```
Sun Aug 19 13:21:58 EDT 2001
Sun Aug 19 13:33:15 EDT 2001
```

Всего 677 секунд. Задействовав кэширование каталогов, мы сберегли 4 секунды. Почему так мало?

Что ж, назначение кэширования – повторное использование чего-либо. Когда вы собираете ту или иную программу, процесс посещает каждый каталог только один раз. Если вы никогда не возвращаетесь на то же место, то использование кэша бессмысленно. Минимальная экономия времени в тех редких случаях, когда `make buildkernel` посещает каталоги повторно, – это все, чего мы добились. Более значительные улучшения можно заметить в случае веб-сервера, который постоянно обращается к одним и тем же файлам.

Итак, наш диск по-прежнему является узким местом. Пришло время для решительных мер.

Перемещение /usr/obj

Процесс `make buildkernel` читает файлы в */usr/src* и записывает их в */usr/obj*. Возьмем один из наших древних дисков SCSI и смонтируем его в */usr/obj*,

оставив /*usr*/*src* на новом диске IDE. В первом тесте применим монтирование без Soft Updates – просто в качестве иллюстрации. Сейчас наши системные диски выглядят так:

```
# df
Filesystem  1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad4s1a    248111    74081   154182    32%    /
/dev/ad4s1f   2032839   133492  1736720     7%    /test1
/dev/ad4s1g   2032839  1266476   603736    68%    /test2
/dev/ad4s1h  29497862  3842891 23295143    14%    /usr
/dev/ad4s1e   3048830   977220  1827704    35%    /var
procfs              4        4        0   100%    /proc
/dev/da0s1e   3525041        1  3243037     0%    /usr/obj
#
```

Теоретически ввод и вывод распределяются между различными дисками. Вывод top подобен предыдущему, однако вывод vmstat выглядит иначе:

```
 procs      memory      page                    disks     faults      cpu
 r b w     avm    fre  flt  re  pi  po  fr  sr ad4 da0   in   sy  cs us sy id
 0 4 0   29200 329436 2433   0   0   0 2109   0   2  12  367 3390 3220 58 24 18
 0 0 0   15388 338376 2298   0   0   0 2939   0   1  23  390 3315 4487 49 24 26
 0 0 0   19124 336604 2453   0   0   0 2559   0   3  30  413 3672 5333 39 29 32
 0 3 0   23680 330252 2000   0   0   0 1818   0  14  65  489 2979 4874 29 24 46
 0 4 0   22832 329136 2628   0   0   0 2768   0   1  16  374 3783 5158 50 28 23
 0 5 0   23404 326976 2624   0   0   0 2702   0   0  15  373 3815 5550 52 29 20
...
```

Процессы по-прежнему блокируются, ожидая доступа к диску. Однако обратите внимание на колонки `ad4` (диск IDE) и `da0` (диск SCSI). Теперь нагрузка распределена между двумя дисками. Временные срезы выглядят так:

```
Sun Aug 19 13:59:54 EDT 2001
Sun Aug 19 14:11:41 EDT 2001
```

Семьсот семь секунд! Так же плохо, как и в начале экспериментов! Ох. Однако вспомним, что этот ветхий диск SCSI без включенных Soft Updates работал так же хорошо, как и современный диск IDE. Покупка современного диска SCSI определенно улучшит производительность. Включим Soft Updates в новом /*usr*/*obj* и посмотрим, что получится:

```
# umount /usr/obj
# tunefs -n enable /usr/obj
tunefs: soft updates set
# mount /usr/obj
#
```

Теперь удалим все в /*usr*/*obj*, перезагрузим систему и снова выполним тест. Вывод vmstat показывает, что дисковая пропускная способность по-прежнему является узким местом. Благодаря Soft Updates период времени снизился до 670 секунд. Soft updates дали нам 6%. Конечно, это не слишком много, однако экономия времени за целый день или при выполнении большой сборки будет ощутимой.

Теперь должно быть ясно, как настраивать систему. Поэкспериментируем с ней еще. Представление */usr/obj* в качестве зеркалированного раздела Vinum с Soft Updates снижает время до 663 секунд, давая 7% улучшение. Это настолько хорошо, насколько возможно. Во всех тестах vmstat показывает, что дисковая пропускная способность является узким местом.

Извлеченные уроки

В предыдущем тестировании мы выяснили, что быстродействие диска, бесспорно, представляет собой узкое место. Четырехлетний диск SCSI обрабатывает данные так же быстро, как и современный диск IDE, однако для достижения лучшей производительности нужен более быстрый диск. Более быстрые диски намного дешевле, чем новые компьютеры – хотя бы потому, что в новых машинах как раз есть быстрые диски.

Лучше всего то, что сейчас вы можете прийти к своему руководителю и сказать: «Это плохой диск. Нам нужен более быстрый. Наш поставщик, *AbsoluteBSD.com*, предлагает их за $400». При этом вы уверены в своих аргументах. Намного хуже, если приходится говорить: «Это плохой сервер; нам нужен новый».

Конечно, кроме `make buildkernel` есть и другие программы, которые предъявляют совершенно иные требования. Их надо оценивать по отдельности. Хотя 5–10% прироста производительности – это не так уж и много, но есть разница между нормальным обслуживанием системы и отчаянными усилиями по установке нового оборудования в срок, чтобы пользователи смогли работать завтра утром.

19

Как обстоят дела?

Каждый системный администратор должен без промедления отвечать на основной вопрос: «Как работает сервер?» Вы должны знать, чем занимается система, чтобы принимать взвешенные решения по планированию и пропускной способности. Ваш руководитель хочет получать отчеты о состоянии систем. Даже если он не спрашивает об этом сейчас, в будущем обязательно спросит. Лучше получить эту информацию до того, как она потребуется.

FreeBSD поддерживает генерацию отчетов, позволяет создавать информативные диаграммы, выполнять ежедневные проверки состояния и т. д. Рассмотрим все это подробнее.

Письма о состоянии

Если заглянуть в */etc/crontab*, то запустив команду periodic(8), можно понять, что FreeBSD выполняет задания по обслуживанию системы каждый день, неделю и месяц. Эти задания выполняют базовую проверку системы и уведомляют администратора об изменениях, компонентах, требующих внимания, и потенциальных «дырах» в защите. Вывод запланированных заданий ежедневно посылается пользователю root на локальной системе. Для того чтобы определить, чем занимается система, проще всего прочитать эти письма; многие системные администраторы, как и вы очень занятые, постарались, чтобы эти сообщения были полезными и необходимыми.

Конфигурация ежедневных, еженедельных и ежемесячных отчетов определяется в */etc/periodic.conf*, */etc/defaults/periodic.conf* и */etc/periodic*. Подробные сведения можно получить в главе 9, а можно сразу перейти к рассмотрению сценариев в */etc/periodic*.

Перенаправление отчетов

Поскольку вы наверняка не хотите каждый день регистрироваться под именем root на всех своих серверах, посылайте свою почту в централизованный почтовый ящик. Для этого надо отредактировать файл */etc/mail/aliases* (см. главу 14), задав перенаправление писем в какой-нибудь другой почтовый

ящик. Если у вас много серверов, то в почтовом ящике будет скапливаться много писем, однако по мере приобретения опыта вы научитесь бегло просматривать отчеты, обращая внимание только на критичные или необычные изменения.

Протоколирование с помощью syslogd

Система протоколирования – одна из восхитительных особенностей UNIX. В отличие от других операционных систем, ограничивающих вас небольшим диапазоном протоколов, система UNIX позволяет протоколировать почти все и почти с любым уровнем детализации. В UNIX предусмотрены приемы протоколирования для основных ресурсов, причем администраторы могут выбрать ту конфигурацию протоколирования, которая соответствует их нуждам. Почти все программы интегрированы с системой протоколирования syslogd (8).

syslogd обслуживает сообщения в соответствии с тем или иным источником сообщений (facility) и уровнем важности (протоколирования). Обе характеристики указываются в сообщениях, направляемых программе syslogd. Для управления системными протоколами надо понимать, что такое источник и уровень, поэтому они будут рассмотрены прямо сейчас.

Источники

Средство[1] (*facility*) – это источник протокольных сообщений (программа), которая отправляет сообщение программе syslogd. У средства может быть произвольное обозначение, по существу, представляющий собой текстовую строку, которая используется для различения программ. В большинстве случаев каждой программе, требующей уникального протокола, необходимо уникальное средство. Многие программы и протоколы имеют средства, выделенные специально для них. Например, FTP – как раз такой типичный протокол, для которого у syslogd есть специальное средство. Кроме того, syslogd поддерживает множество общих средств, которые могут быть задействованы любой программой. Программы могут ничего не говорить о своем средстве, однако их можно отследить по имени программы, указываемом в сообщении.

Ниже перечислены стандартные средства и тип информации, которую они предоставляют.

`auth`

Общедоступная информация, относящаяся к авторизации пользователей, например login и su.

---

[1] Практически везде facility в данном контексте переводят словом «средство». Мы постараемся избежать этого, чтобы смысл текста был понятнее, ибо под «facility» здесь понимается именно источник сообщения. Конечно, читатель, наделенный воображением, может представить себе, что facility – это средство передачи сообщения syslogd, но нам такая интерпретация представляется замысловатой. – *Примеч. науч. ред.*

`authpriv`

Секретная информация, относящаяся к авторизации пользователей. Доступ к ней разрешен только root и другим избранным пользователям.

`console`

Сообщения, обычно выдаваемые на системную консоль, могут быть собраны с помощью средства console.

`cron`

Сообщения от диспетчера системы проходят через средство cron.

`daemon`

Вместилище для сообщений всех системных демонов, не имеющих других обработчиков в syslogd или своей программе.

`ftp`

Демоны FTP будут автоматически отправлять сообщения этому средству (см. главу 15).

`kern`

Сообщения от ядра.

`lpr`

Сообщения от системы печати.

`mail`

Сообщения от почтовой системы.

`mark`

Не является сообщением о событии; средство mark просто помещает запись в ваш протокол каждые 20 минут. Это полезно при комбинировании с другим протоколом.

`news`

Ловит сообщения от демонов Internet News.

`ntp`

Собирает сообщения от синхронизирующего сетевого протокола (Network Time Protocol).

`security`

Собирает сообщения от различных программ защиты, таких как ipf(8) и ipfw(8).

`syslog`

Сервис протоколирования может протоколировать самого себя. Однако не выполняйте протоколирование, когда получаете сообщения от syslogd. В противном случае может произойти путаница.

`user`

Вместилище пользовательских сообщений. Если вы не указали средство протоколирования для пользовательских программ, они задействуют это средство.

`uucp`

Протоколирует UNIX-to-UNIX Copy Protocol. (Этот протокол применялся в UNIX до эпохи Интернета. Вероятно, вы с ним никогда не столкнетесь.)

`local0 - local7`

Зарезервировано для нужд администратора. Во многих программах можно задавать средство протоколирования. Выберите одно из этих средств, если такая возможность предоставляется. Например, сервис поддержки пользователей может протоколироваться в local0.

Уровни

Уровень протоколирования представляет относительную важность сообщений. Хотя программы посылают все данные протоколирования демону syslogd, большинство систем записывают только важную информацию и отбрасывают незначительные сообщения. Конечно, информация, которую один администратор считает пустяковой, может быть жизненно важной для другого. Именно здесь на помощь приходят уровни.

FreeBSD предлагает восемь уровней важность syslogd. С помощью этих уровней можно извещать syslogd о том, что надо записывать, а что следует отбросить.

`emerg`

В системе авария. Сообщения поступают на каждый терминал. По существу, система находится в аварийном состоянии. Вам даже не надо перезагружать систему – система сделает это за вас.[1]

`alert`

Дела в системе плохи, однако не настолько плохи, как на уровне emerg. Система может продолжать работать, однако этой ошибке следует немедленно уделить внимание.

`crit`

Это критические ошибки, например проблемы с аппаратными средствами (скажем, сбойные блоки или поврежденный кабель SCSI) или серьезные трудности с программным обеспечением. Если вы смелы, можете продолжать работу.

`err`

Разнообразные ошибки. Это непорядок. Такие ошибки следует исправлять, однако они не разрушают систему.

---

[1] Это может показаться вам забавным, но только не в том случае, если произойдет с вами.

`warning`

Различные предупреждения.

`notice`

Общая информация, которую можно протоколировать при необходимости. Однако эта информация не требует действий с вашей стороны.

`info`

Общая программная информация, например данные об отдельных транзакциях почтового сервера.

`debug`

Этот уровень предназначен для программистов. Изредка его выбирают системные администраторы, которые пытаются выяснить, почему программа выполняется так, а не иначе. Протоколы отладки могут содержать всю информацию, необходимую программисту для отладки кода. Здесь же может оказаться конфиденциальная информация о пользователях.

`none`

Это означает «ничего не протоколировать с помощью этого средства». Чаще всего применяется при исключении информации из групповых записей (wildcard entries). Эта возможность будет рассмотрена далее.

Файл syslog.conf

В файле */etc/syslog.conf* есть две колонки. В первой описана информация, подлежащая протоколированию. Здесь указывается источник и уровень. Во второй приводится действие, которое будет выполнено, если протокольное сообщение соответствует описанию. syslogd сравнивает каждое предложение с записями в */etc/syslog.conf*. Когда найдена соответствующая запись, выполняется предписанное действие.

Информационные источники представлены источником и уровнем, которые разделены точкой. Когда вы указываете уровень, система по умолчанию будет записывать сообщения этого и более высоких уровней. Рассмотрим следующую запись */etc/syslog.conf*:

```
mail.info                /var/log/maillog
```

Эта запись предписывает системе протоколировать сообщения от почтовой системы в */var/log/maillog*, если уровень важность равен или выше `"info"`.

Групповые символы

При указании информационного источника можно применять групповые символы. Например, следующая строка предписывает протоколирование абсолютно всех сообщений от почтовой системы:

```
mail.*                   /var/log/maillog
```

Для протоколирования всех сообщений из любых источников раскомментируйте запись `all.log`:

```
*.*                    /var/log/all.log
```

Результат вы получите, но информации будет слишком много, чтобы ее можно было применить на практике. Для нахождения необходимых фрагментов придется выполнять сложные команды grep. Кроме того, в протоколы попадет вся секретная информация, что обусловлено уровнем debug.

Исключение информации

Применяйте средство authpriv и уровень none для исключения аутентификационной информации. Точка с запятой применяется для комбинирования записей в одной строке:

```
*.*;authpriv.none         /var/log/all.log
```

Сравнение

Операторы сравнения < (меньше), = (равно) и > (больше) также можно применять в */etc/syslog.conf*. Их можно указывать вместе с уровнями и протоколировать данные выше определенного уровня в один файл, а данные ниже определенного уровня – в другой. По умолчанию syslogd записывает все сообщения указанного или вышестоящих уровней, но можно задать диапазон необходимой информации.

Для примера предположим, что вам нужен один протокол для почтового трафика и другой – для информации, выдаваемой при отладке почтовой службы:

```
mail.info           /var/log/maillog
mail.=debug         /var/log/maillog.debug
```

Запись mail.info собирает все протокольные сообщения, посылаемые средству mail, которые соответствуют уровню info или вышестоящему уровню. Вторая строка охватывает только сообщения уровней не выше debug. Нельзя использовать источник сообщений mail.debug, иначе протокол отладки будет содержать все записи предыдущего протокола! Таким образом, необязательно просматривать отладочную информацию, чтобы узнать о деятельности почтового сервера. Кроме того, не надо пробираться сквозь дебри информации о передаче почты, чтобы получить отладочный вывод почтового сервера.

Локальные средства

Во многих программах предусмотрена возможность использовать syslogd для протоколирования. Большая их часть позволяет выбирать необходимое средство. Для таких случаев зарезервированы различные локальные средства.

Например, можно предписать программе протоколировать события с помощью средства local3. Способ использования средств варьируется от программы к программе. Выбрав программу, которая будет обозначать сообщения в соответствии с некоторым средством и посылать их демону syslogd, можно предписать syslogd перехватывать эти сообщения и что-либо делать с ними.

```
local3.*                /var/log/programname
```

В целом, если программа поддерживает протоколирование с помощью средства, применяйте локальное средство.

Протоколирование по имени программы

Если в системе нет средств или программа не поддерживает syslogd, то для протоколирования можно задействовать имя программы. Для записи с именем требуются по крайней мере две строки: имя программы, которому предшествует знак восклицания, и строка с информацией о протоколировании.

Например, для протоколирования ppp подойдет такая запись:

```
!ppp
*.*                     /var/log/ppp.log
```

В этой записи сначала указывается имя программы, а затем с помощью групповых символов демону syslogd предписано добавлять в файл абсолютно все сообщения. (Неизвестно, насколько разумный уровень протоколирования предлагает та или иная программа, поэтому надежнее всего записывать абсолютно все, пока не выяснятся особенности протоколирования данной программы.)

Хост для протоколирования

Обычно в сетях я предусматриваю отдельный хост для протоколирования, обслуживающий не только машины FreeBSD, но и маршрутизаторы Cisco, коммутаторы 3Com, машины с другими UNIX и любые другие системы, умеющие общаться с syslogd. Наличие одного хоста, на котором ведется все протоколирование, упрощает обслуживание систем.

Для отправления протокольных собщений другому хосту применяется символ @. Например, следующая строка задает выполнение дампа всех сообщений, предназначенных syslogd, на хост протоколирования в моей сети:[1]

```
*.*             @loghost.absolutebsd.com
```

Файл */etc/syslog.conf* на хосте протоколирования определяет окончательное место назначения для сообщений, которые получает хост. К счастью, в каждом протокольном сообщении есть имя хоста.

Протоколирование пользовательских сессий

Для протоколирования пользовательских сессий укажите имена пользователей через запятую. Тогда поступающие протокольные сообщения будут отображаться на терминалах указанных пользователей.

Для отображения сообщений на терминалах всех пользователей укажите звездочку (*) в качестве места назначения. Например, в файле *syslog.conf* по умолчанию есть такая строка:

---

[1] Не следуйте бездумно этому примеру, иначе я найду интересные сведения в ваших протоколах и размещу их на известных общедоступных серверах. Не выполняйте протоколирование на чужом хосте!

```
*.emerg                    *
```

Данная запись говорит, что любое сообщение уровня emerg (emergency, авария) появится на терминалах всех пользователей.

Отправка протокольных сообщений программам

Наконец, если необходимо, чтобы протоколы обрабатывала другая программа, применяйте символ конвейеризации (|) для перенаправления сообщений этой программе:

```
mail.*              |/usr/local/bin/mailstats.pl
```

Примечание

По традиции системы UNIX требуют наличия символов табуляции между колонками */etc/syslog.conf*, однако FreeBSD разрешает применять пробелы. Символы табуляции следует использовать лишь при разделении одного *syslog.conf* между различными системами UNIX.

Ротация протоколов с помощью newsyslog.conf

Файлы протоколов растут, и их рост надо контролировать. Стандартный способ – *ротация протоколов (log rotation)*. При ротации протоколов самые старые протоколы удаляются. Каждому старому протоколу назначается имя следующего по старшинству, текущий протокол перемещается и создается новый протокол.

В состав FreeBSD входит базовый обработчик протоколов newsyslog(8), способный также сжимать файлы, перезапускать демоны и выполнять все рутинные операции по ротации протоколов. cron запускает newsyslog ежечасно.

newsyslog считывает */etc/newsyslog.conf* и проверяет каждый файл протокола, указанный в конфигурационном файле. Если условия ротации протокола выполняются, то происходит ротация протокола и другие надлежащие действия.

В файле */etc/newsyslog.conf* каждому протоколу отводится одна строка с семью полями. Например:

```
/var/log/slip.log     root:network    640  3     100  *     Z
```

Поочередно рассмотрим каждое поле.

Путь к файлу протокола

В первом поле каждой строки содержится полный путь к обрабатываемому файлу протокола (в данном примере /var/log/slip.log).

Владелец и группа

Второе поле (в данном примере root:network) указывает владельца и группу обрабатываемого файла, которые разделены двоеточием (скажем, root:wheel). Это необязательное поле, отсутствующее во многих стандартных записях.

newsyslog может изменять владельца и группу старых файлов протокола. По умолчанию файлами протокола владеет root, и они принадлежат группе wheel. Обычно владельца этих файлов не меняют, но к этой возможности имеет смысл прибегнуть на многопользовательских машинах.

Можно изменять либо владельца, либо группу. В данных случаях необходимо применять точку с запятой, даже если перед или после этого символа ничего нет. Например, значение `:www` изменит группу на www, а значение `user827:` изменит группу на user827.

Права доступа

Третье поле (`640` в данном примере) – это права доступа в стандартном для UNIX представлении из трех цифр. (Для получения подробных сведений обратитесь к главе 7.)

Количество

Следующее поле – *количество* (*count*); в данном примере его значение равно 3. Оно представляет количество старых файлов протокола, которые хранит newsyslog. Подсчет архивированных файлов протоколов начинается с 0. Во многих компьютерных системах нумерация начинается с 0, однако newsyslog добавляет 0 к значению счетчика. Если значение по умолчанию, заданное для */var/log/messages*, равно 5, то в каталоге */var/log* окажутся следующие файлы:

```
messages
messages.0.gz
messages.1.gz
messages.2.gz
messages.3.gz
messages.4.gz
messages.5.gz
```

Те из вас, кто умеет считать, обнаружат наличие пяти, а не шести резервных копий плюс текущий файл протокола! Как правило, лучше иметь больше файлов протоколов, чем столкнуться с их нехваткой. Если в системе мало дискового пространства, то удаление одного-двух файлов выручит вас на некоторое время. Некоторые веб-серверы могут вмещать сотни сайтов; удаление одного-двух файлов для каждого сайта может высвободить много дискового пространства.

Размер

Пятое поле (в данном примере 100) – это размер файла. При запуске программа newsyslog сравнивает размер, указанный в этом поле, с размером файла. Если размер файла больше указанного, файл подвергается ротации. Если размер файла не должен влиять на проведение ротации, поставьте здесь звездочку (*).

Время

Пока что все просто, верно? Что ж, шестое поле, время (* в данном примере), может доставить хлопот начинающим администраторам.

В поле времени возможно четыре значения: звездочка (*), число и дата в двух различных форматах. Если вы не хотите проводить ротацию протоколов в определенное время, поместите здесь звездочку (*). Если поставить простое число, newsyslog осуществит ротацию протоколов по истечении заданного количества часов. Например, если необходимо проводить ротацию каждые 24 часа и неважно, когда именно она происходит, укажите в этом поле значение 24.

Формат даты немного более сложен.

Формат времени ISO8601

Любая запись, начинающаяся с символа @, – это время в ограниченном формате ISO 8601. Это стандартный формат, применяемый программой newsyslog в большинстве систем UNIX. Первоначально данный формат применялся в базовой программе newsyslog, разработанной в MIT. К сожалению, этот стандарт не совсем ясен, однако раз это стандарт, FreeBSD его поддерживает.

Полная дата в формате ISO 8601 образована 16 цифрами с символом T в середине. Первые четыре цифры означают год, следующие две – месяц, следующие две – дату. Символ T после даты играет роль точки в десятичной дроби, отделяя целые дни от доли одного дня. Этот символ должен присутствовать в дате формата ISO 8601.

Следующие две цифры – это часы, следующие две – минуты, следующие две – секунды. Например, дата 2 февраля 2002 года и время 21 час 15 минут и 8 секунд в формате ISO 8601 будут представлены так:

```
20020202T211508
```

Даты в формате ISO 8601, приведенные полностью, достаточно просты, однако при неполном указании даты возникает путаница. Например, если указать только поля рядом с T, а остальные оставить пустыми, то незаполненные поля будут расценены как групповые символы. Например, T23 соответствует 23 часам каждого дня года. При указании @T23 протоколы будут подвергаться ротации каждый день в 11 часов вечера, а 4T00 соответствует полночи 4 числа каждого месяца.

Как и в случае crontab, необходимо указать часы. Дата, подобная @7T, означает ротацию протоколов каждый час в седьмой день месяца. В конце концов, такая запись соответствует целому дню! Это полезно при отладке, но в целом нежелательно.

Формат времени, специфичный для FreeBSD

В формате ISO 8601 нехорошо то, что он не позволяет легко назначать еженедельные задания. (Например, нет ничего необычного в ротации протоколов по понедельникам.) Кроме того, невозможно просто указать последний

день месяца. Именно здесь на помощь приходит другой формат времени. Любое время, которому предшествует знак доллара ($), представлено в формате *месяц-неделя-день*, специфичном для FreeBSD.

Этот формат подобен записи cron. Он позволяет устанавливать конкретные дни недели, в которые надлежит запускать задания. В нем применяются три идентификатора: `M` (день месяца), `W` (день недели) и `H` (час дня). За каждым идентификатором следует число, означающее конкретное время выполнение задания. Часы задаются в диапазоне от 0 до 23, а дни недели – от 0 (воскресенье) до 6 (суббота). Нумерация `M` начинается с 1 и доходит до количества дней в конкретном месяце. Например, для ротации протоколов каждое воскресенье в 8 часов утра надо указать `$W0H8`. Для ротации протоколов пятого числа каждого месяца в полдень укажите `$M5H12`.

Интересная особенность такой системы состоит в том, что она позволяет автоматически запускать задание в последний день месяца. Для этого надо задать символ `L`. Не будь такой возможности, пришлось бы создавать сценарий, который знает количество дней в каждом месяца, сравнивает текущую дату с запланированной и принимает решение о запуске программы. (Это быстро надоест.) Например, для запуска анализатора файлов протоколов за два часа до истечения месяца применяйте запись `$MLH22`.

Примечание

Ротацию можно проводить в заданное время, или по достижении определенного размера протоколов, или в том и другом случае. Если указать оба варианта, то протокол будет подвергнут ротации при выполнении любого из условий. Если ротация необходима лишь при выполнении одного из условий (то есть необходимо проводить ротацию каждый день вне зависимости от размера файла), поставьте в другом поле звездочку (*).

Флаги

Теперь, когда вы знаете, как выражать точное время при протоколировании, рассмотрим поле флагов (в данном случае его значение равно `Z`). Это поле предлагает два варианта обслуживания файлов протоколов. Одни программы протоколируют данные в виде простого текста, другие – в двоичном формате; протоколы этих видов надо воспринимать по-разному.

Двоичные файлы можно записывать только специальным образом. Каждый новый файл протокола программа newsyslog начинает с сообщения «log-file turned over» (файл протокола подвергнут ротации). Флаг `B` сообщает программе newsyslog, что это двоичный файл и сообщение не следует записывать. Однако многие файлы протоколов представляют собой простой текст ASCII, поэтому их сжатие позволит сохранить много свободного места на диске. Флаг `Z` предписывает программе newsyslog сжимать файлы протоколов с помощью gzip.

Можно задать только один из этих флагов – в конце концов, сжатие двоичных файлов не сбережет много пространства, а сообщение о ротации можно записывать только в текстовые протоколы.

Путь к pid-файлу

Следующее поле – путь к *pid-файлу* (в данном примере не показано). pid-файл – это простой способ записи идентификатора процесса (PID) той или иной программы, причем эту запись могут легко просматривать другие программы. Такие файлы есть не у всех программ, а те программы, у которых они есть, хранят эти файлы в */var/run* (загляните в этот каталог и посмотрите, что есть в вашей системе).

Если вы приводите полный путь к pid-файлу в */var/run*, то при ротации протоколов newsyslog пошлет программе один из сигналов kill. Например, веб-сервер Apache надо уведомить о том, что его протоколы подвергаются ротации. Если указать здесь pid-файл, newsyslog пошлет серверу Apache сигнал `kill -1`, чтобы он обслужил ротацию протоколов.

Большинство программ обслуживают ротацию по сигналу `kill -1` или `SIGHUP`, однако некоторым программам при ротации протоколов нужен специальный сигнал. Для таких программ в последнем поле можно указать точный номер сигнала.

Пример записи newsyslog.conf

Собрав все это вместе, рассмотрим худший случай – пример «вы-*должно-быть*-шутите». Предположим, у вас есть протокол базы данных, ротацию которого необходимо проводить в 11 часов вечера последнего дня каждого месяца. В документации базы данных сказано, что для ротации файлов требуется послать программе сигнал прерывания (`SIGINT`, или сигнал с номером 2). Вам необходимо, чтобы архивированными протоколами владел пользователь dbadmin. Просмотр протоколов разрешен только этому пользователю, а хранить протоколы надо шесть месяцев. Более того, протоколы представляют собой двоичные файлы. Строка файла *newsyslog.conf* будет такой:

```
/var/log/database   dbadmin:  600  6    *    $MLH23 B /var/run/db.pid  2
```

Этот пример – исключение; в большинстве случаев достаточно указать имя файла и условия ротации.

Отчеты с помощью SNMP

Письма с отчетами полезны, однако в них содержится только общая информация, а протоколы, относящиеся к продолжительным периодам времени, трудно читать. Чтобы узнать больше о работе сервера сейчас и в долгосрочной перспективе, применяйте простой протокол управления сетью (Simple Network Management Protocol, SNMP). SNMP – это индустриальный стандарт для сбора информации о хостах сети. Данный протокол поддерживают большинство поставщиков, а представленные здесь методы можно применять для мониторинга любого устройства, понимающего SNMP.

Существуют бесплатные инструменты, которые предписывают SNMP генерировать замечательные отчеты о работе системы. Однако для эффективного применения этих инструментов необходимо понимать, что собой представ-

ляет SNMP и как он работает. Рассмотрев основы SNMP, обсудим наиболее популярные инструменты SNMP во FreeBSD.

Основы SNMP

SNMP работает по стандартной модели «клиент/сервер». Клиент SNMP, или агент, посылает по сети запрос серверу SNMP. Сервер SNMP, snmpd, собирает информацию с локальной системы и возвращает ее клиенту.

Агент SNMP может также посылать серверу SNMP запрос на выполнение изменений. Если система сконфигурирована надлежащим образом (или ненадлежащим, в зависимости от вашей точки зрения), вы можете выдавать команды через SNMP. Такая конфигурация с помощью «записи» чаще всего применяется в маршрутизаторах, коммутаторах и других сетевых устройствах.

В UNIX есть собственная система конфигурирования, которая обычно не позволяет выдавать команды через SNMP. (Некоторые демоны можно конфигурировать через SNMP. Можно написать сценарии командной оболочки, устанавливающие значения SNMP, но это специальные случаи.) Написание таких конфигураций выходит за рамки темы этой книги. Создание системных конфигураций и команд, выдаваемых через SNMP, требуют тщательной настройки и внимания к безопасности, и это отличная тема для целой книги.

MIB

SNMP управляет информацией через базу управляющей информации (Management Information Base, MIB). (Пример дерева MIB представлен в главе 4 – там, где рассмотрены sysctls.)

Каждый сервер SNMP имеет список информации, которую он может извлечь из локального компьютера. Сервер представляет эту информацию в виде иерархической системы, или дерева MIB. Каждое дерево MIB имеет очень общую основную категорию. Кроме того, в дереве MIB представлены сеть, физическая система, программы и т. д. Категории разбиты на более специфичные подразделы. Такие деревья представляют собой хорошо организованные каталоги, подкаталоги которых содержат более специфичную информацию. Аналогично MIB верхнего уровня содержат множество MIB более низких уровней.

На MIB ссылаются по имени и по номеру. Например, так выглядит MIB одной из систем:

```
interfaces.ifTable.ifEntry.ifOutErrors.1
```

Первый элемент этого MIB, `interfaces`, показывает нам, что запись относится к сетевым интерфейсам системы (сетевые карты, параллельные порты и т. д.). Если бы в данной системе не было интерфейсов, первая категория даже не существовала бы. Поле `ifTable` – это *таблица интерфейсов*, или список всех интерфейсов системы. Поле `ifEntry` показывает один конкретный интерфейс, а `ifOutErrors` означает исходящие ошибки на конкретном интерфейсе.

Наконец, замыкающий символ 1 означает, что мы заинтересованы в интерфейсе 1.

MIB можно также выражать в виде номеров. Большинство инструментов SNMP отдают предпочтение числовым MIB. К сожалению (или нет), наши немощные мозги предпочитают слова, однако они должны уметь[1] работать и с тем и с другим. (Описание такого преобразования будет рассмотрено в разделе «Преобразование номеров и имен», а пока поверьте мне на слово.) Предыдущий пример можно преобразовать в такую форму:

```
.1.3.6.1.2.1.2.2.1.20.1
```

Если выразить этот параметр MIB в словах, то он состоит из пяти элементов, разделенных точками. MIB, выраженный в числах, состоит из 11 компонентов. Ощущение такое, что если обе записи должны означать одно и то же, то это неправильно. В чем тут дело?

Числовой MIB длиннее, поскольку в него входит часть по умолчанию – `.1.3.6.1.2.1`. Она означает `.iso.org.dod.internet.mgmt.mib-2`. Это стандартное подмножество MIB, применяемое в Интернете (SNMP также можно задействовать для управления устройствами, не относящимися к Интернету). В названии почти каждого MIB есть префиксная строка – вот почему никто ее больше не приводит.

Если потянуло на чудачества, можете даже смешать слова и числа:

```
.1.org.6.1.mgmt.1.interfaces.ifTable.1.ifOutErrors.1
```

Эти MIB выглядят полезными, но как определить назначение каждого из них? SNMP MIB могут изменяться от устройства к устройству и от сервера к серверу. Обратитесь к документации вашего агента SNMP или вашего устройства и посмотрите, какие MIB доступны.

При выполнении запроса SNMP вы используете MIB для извлечения именно той информации, которая вам необходима. Поскольку большинство сетевых устройств не предоставляют информацию кому попало, рассмотрим, как в SNMP организована защита с помощью сообществ.[2]

net-snmp

net-snmp (*http://net-snmp.sourceforge.net*) – это лучший набор программ SNMP, которые работают на FreeBSD. Он мал, открыт для расширения и эффективен. В состав FreeBSD он входит в виде «порта» (*/usr/ports/net/net-snmp4*). Это популярный пакет, а «порт» обычно не отстает от последних изменений, поскольку разработчики net-snmp сильно заинтересованы во FreeBSD. Они от-

---

[1] *Удовольствие* при этом отнюдь не гарантируется, оно лишь возможно.

[2] Сообщество SNMP – это группа, к которой принадлежат компьютеры (и возможно, какие-то еще устройства) с сервисом SNMP. Агент не примет запрос от системы управления, сконфигурированной за пределами его сообщества. – *Примеч. науч. ред.*

зываются на сообщения о проблемах, запросы о помощи и (еще лучше) подготавливают заплатки.

Примечание

Небольшие трудности вызывает тот факт, что пакет net-snmp раньше назывался ucd-snmp. Калифорнийский университет в Дэвисе больше не вовлечен в этот проект, поэтому название изменено. В различных источниках встречаются ссылки как на net-snmp, так и на ucd-snmp.

При установке net-snmp из «порта» вам будут заданы различные вопросы. Принимайте значения по умолчанию; их можно будет изменить с помощью файла конфигурации, который мы создадим далее.

В состав «порта» net-snmp входит клиент SNMP, сервер SNMP и инструмент преобразования. Рассмотрим каждый из них по отдельности.

snmpwalk

«Порт» net-snmp включает в себя настраиваемый инструмент, snmpwalk(1), позволяющий исследовать дерево SNMP на других хостах. snmpwalk хорошо работает на серверах всех видов. В своей системе FreeBSD я применяю snmpwalk для исследования маршрутизаторов Cisco, разнородных коммутаторов, других машин BSD и даже систем Windows. Для применения snmpwalk укажите имя хоста, который необходимо проверить, и имя сообщества SNMP:

```
# snmpwalk hostname community
```

Попробуйте snmpwalk в системе, на которой запущен демон SNMP, указав сообщество по умолчанию, `"public"`:

```
# snmpwalk localhost public > snmpwalk.out
#
```

Объем информации, выводимой командой snmpwalk, велик, поэтому в данном примере мы перенаправим его в файл. Изучите его.

Специальные запросы snmpwalk

Через SNMP также можно выполнять специальные запросы, просто указав интересующую вас часть дерева. Предположим, требуется узнать значение MIB `.1.3.6.1.4.1.2021.11.9.0`. (Это процент времени CPU, затраченного на пользовательские программы при запуске сервера net-snmp. В следующем разделе рассказано, как получить эту информацию.) Если MIB добавить к концу запроса snmpwalk, то будут получены сведения именно о нем.

```
#snmpwalk localhost public .1.3.6.1.4.1.2021.11.9.0
enterprises.ucdavis.systemStats.ssCpuUser.0 = 1
#
```

Как следует из вывода команды, на запуск пользовательских программ потрачен 1% времени CPU. Здорово, правда?

Программу snmpwalk можно применять для получения информации от любого устройства, которое понимает SNMP. Например, документация Windows NT сообщает, что MIB 1.3.6.1.4.1.311.1.1.3.1.1.1.1 представляет «доступную память». Как с помощью snmpwalk в системе FreeBSD проверить это значение, не регистрируясь в системе NT? Попробуйте так:

```
# snmpwalk windowsserver public .1.3.6.1.4.1.311.1.1.3.1.1.1.1.0
enterprises.311.1.1.3.1.1.1.1.0 = 154447872
#
```

Специальные запросы выполнять намного проще, нежели регистрироваться в системе и запускать top; кроме того, такие запросы можно выполнять в любой системе с инструментами SNMP. Позже мы будем автоматически опрашивать серверы по MIB и применим результаты для создания графиков состояния серверов.

Коммерческие продукты для проверки серверов SNMP стоят сотни и тысячи долларов. Применение бесплатных инструментов SNMP – отличный первый шаг к использованию FreeBSD в любой сети.

Преобразование номеров и имен

Итак, вы хотите знать, как переходить от номеров к именам и обратно. Вам интересно, какие MIB доступны в вашей системе. Возможно, вы желаете знать, что они означают. Что ж, это *вполне* разумные вопросы. В состав «порта» net-snmp входит инструмент преобразования snmptranslate(1).

Прежде всего, выберите MIB, обозначение которого вы хотите преобразовать. Вы ведь сохранили вывод команды snmpwalk? Что ж, возьмем фрагмент вывода и изучим его. Следующая строка взята из моей локальной системы:

```
enterprises.ucdavis.memory.memAvailSwap.0 = 204672
```

Строка выглядит весьма значительно, не так ли? О доступном пространстве свопа полезно знать.

Для начала надо уяснить, что snmpwalk выдает только последнюю часть MIB. Вы должны знать, что дерево enterprises всегда предваряется .1.3.6.1.4. (В мире SNMP это общеизвестно.) Теперь, вооружившись этими тайными знаниями, зададим для snmptranslate полный MIB и ключ -Td:

```
# snmptranslate -Td .1.3.6.1.4.enterprises.ucdavis.memory.memAvailSwap.0
.1.3.6.1.4.1.2021.4.4.0
memAvailSwap OBJECT-TYPE
  -- FROM       UCD-SNMP-MIB
  SYNTAX        INTEGER
  MAX-ACCESS    read-only
  STATUS        current
  DESCRIPTION   "Available Swap Space on the host."
::= {  iso(1) org(3) dod(6) internet(1) private(4) enterprises(1) ucdavis(2021)
memory(4) memAvailSwap(4) 0 }
#
```

Выдана масса полезной информации о MIB, включая его числовой эквивалент (`.1.3.6.1.4.1.2021.4.4.0`) и его описание (`"Available Swap Space on the host."`, доступное пространство свопа в хосте). Обратите внимание на числовой MIB; вскоре он понадобится. Последний фрагмент представляет собой имя каждой части MIB и соответствующий ей номер.

Примечание

Эту команду также можно запустить наоборот, задав числовой MIB. В результате будет выдана полная расшифровка имени MIB.

Настройка snmpd

Прежде чем применять SNMP для мониторинга системы, надо настроить демон snmpd. Вместо того чтобы непосредственно установить конфигурационный файл по умолчанию, «порт» net-snmp устанавливает предопределенный конфигурационный файл в */usr/local/share/examples/ucd-snmp/EXAMPLE.conf*. В дополнение к предопределенным сообществам «public» и «private» в конфигурации по умолчанию есть множество возможных прорех в защите.

К счастью, в состав net-snmp входит программа для создания конфигурационного файла демона SNMP, snmpconf(1). Для создания стандартной конфигурации с помощью snmpconf запустите такую команду:

```
# snmpconf -i -g basic_setup
```

Эта команда проведет вас по этапам базового конфигурирования snmpd, установив файл настройки в надлежащем месте. (Поскольку мы запустим snmpconf только один раз, то не будем внимательно рассматривать каждую часть этой команды.) Конфигурирование особой сложности не представляет, однако мы пройдем по его этапам, чтобы все установить правильно. (Хотя эта программа позволяет установить различные дополнительные параметры, мы будем конфигурировать только необходимые, чтобы выполнять базовый мониторинг.)

snmpconf задаст много вопросов, на которые нужно будет ответить «yes» или «no». Сначала должны появиться такие строки:

```
************************************************
*** Beginning basic system information setup ***
************************************************
Do you want to configure the information returned in the system MIB group
(contact info, etc)? (default = y):
(Начало настройки базовой системной информации
Хотите ли вы конфигурировать информацию, возвращаемую в группу системных MIB
(контактная информация и т. д.)? (значение по умолчанию = y):
```

Вы действительно хотите предоставить базовую системную информацию, поэтому наберите y.

```
Configuring: syslocation      (Конфигурирование: расположение системы)
Description:                  (Описание)
```

```
  The [typically physical] location of the system.
                                    (Расположение системы (обычно физическое))
    arguments:  location_string     (аргументы: location_string)

The location of the system:         (Расположение системы:)
```

Введите здесь расположение системы, например «серверная комната», «серверная комната А-30» или что-либо другое. Такое обозначение может быть полезно при наличии множества серверов в различных местах; в случае одного сервера в этом нет смысла. Однако будьте внимательны: введенная информация будет показана при мониторинге, поэтому не размещайте здесь ничего обидного или неуместного.

Далее сценарий запросит «syscontact» – информацию об администраторе, ответственном за систему. Если у сервера есть несколько системных администраторов, то это замечательное место для размещения электронного адреса администратора, к которому следует обращаться при возникновении проблем или вопросов. Кроме того, здесь можно разместить псевдоним, который обеспечивает перенаправление писем нескольким сотрудникам. В этом случае вам не придется переконфигурировать snmpd, если сотрудник, ответственный за snmpd, уволится или займет другую должность.

```
Do you want to properly set the value of the sysServices.0 OID? (default = y):
(Хотите ли вы должным образом установить значение sysServices.0 OID
 (значение по умолчанию = y):
```

Да, вы хотите. В этом разделе описываются самые базовые свойства системы, однако не те из них, которые принимают ответы «y» или «n». Поскольку для конфигурации SNMP вы будете подбирать действительные значения, введите 0 для «no» и 1 для «yes».

```
Configuring: sysservices                (Конфигурирование: системные сервисы)
Description:                            (Описание)
  The proper value for the sysServices object.
                                   (Правильное значение для объекта sysServices)
    arguments:  sysservices_number      (аргументы)

does this host offer physical services (eg, like a repeater) [answer 0 or 1]:
(предлагает ли этот хост физические сервисы (например, повторитель)
[ответьте 0 или 1]
```

Некоторые администраторы являются обладателями telco-карт для FreeBSD, позволяющих подключать стандартную машину x86 PC к телефонному коммутатору, чтобы обслуживать телефонные операции на уровне компании. Если таких устройств у вас нет, то при ответе на предыдущий вопрос введите 0.

```
does this host offer datalink/subnetwork services (eg, like a bridge):
(предлагает ли этот хост сервисы канала передачи данных или подсетей (например, мост):)
```

FreeBSD можно конфигурировать как мост. Однако если это не сделано, ответьте 0.

```
does this host offer internet services (eg, supports IP): 1
does this host offer end-to-end services (eg, supports TCP): 1
```

```
does this host offer application services (eg, supports SMTP): 1
(предлагает ли этот хост сервисы Интернета (например, поддержку IP): 1
предлагает ли этот хост сервисы доставки (например, поддержку TCP): 1
предлагает ли этот хост прикладные сервисы (например, поддержку SMTP): 1)
```

Ответьте 1 на все три предыдущих вопроса. Сервер Интернета на платформе FreeBSD поддерживает все эти виды сервиса.

```
*************************************
*** BEGINNING ACCESS CONTROL SETUP ***       (Начало настройки управления доступом)
*************************************
Do you want to configure the agent's access control? (default = y):
(Хотите ли вы настраивать управление доступом к агенту (значение по умолчанию = y):)
```

Вы не хотите, чтобы любой желающий мог получить доступ к системе, поэтому ответьте y.

```
Do you want to allow SNMPv3 read-write user based access (default = y):
Do you want to allow SNMPv3 read-only user based access (default = y):
Do you want to allow SNMPv1/v2c read-write community access (default = y):
(Разрешаете ли вы доступ  для чтения-записи SNMPv3 на основе данных
о пользователях (значение по умолчанию = y):
Разрешаете ли вы доступ только для чтения SNMPv3 на основе данных
о пользователях (значение по умолчанию = y):
Разрешаете ли вы доступ для чтения-записи SNMPv1/v2c на основе данных
о сообществах (значение по умолчанию = y):
```

SNMP версии 3 – усовершенствованный протокол, возможности которого выходят далеко за рамки базового мониторинга. Кроме того, мы не используем чтение-запись SNMP; мы хотим только считывать информацию, а не выдавать команды в системе! Ответьте n на три вышеприведенных вопроса. Доступ только для чтения будет получен, если на следующий вопрос ответить y.

```
Do you want to allow SNMPv1/v2c read-only community access (default = y): y
(Разрешаете ли вы доступ только для чтения SNMPv1/v2c на основе данных
   о сообществах (значение по умолчанию = y): y)
Configuring: rocommunity          (Конфигурирование:)
Description:                      (Описание:)
  a SNMPv1/SNMPv2c read-only access community name
  (Название сообщества при доступе только для чтения - SNMPv1/SNMPv2c)
    arguments:  community [default|hostname|network/bits] [oid]      (аргументы:)

The community name to add read-only access for:
(Название сообщества, которому следует разрешить доступ только для чтения:)
```

Администраторам, работавшим с SNMP, знакомы сообщества по умолчанию «public» и «private». Однако применять их не стоит: именно на них первым делом обращает внимание взломщик. Имена сообществ должны быть подобны хорошим паролям; не применяйте названия, которые легко разгадать, или обычные слова, смешивайте буквы и другие символы. Если вы не намереваетесь разрешать кому-либо выполнение команд SNMP, то достаточно одного имени сообщества.

```
The hostname or network address to accept this community name from [RETURN for all]:
(Имя хоста или адрес сети, откуда следует принимать это имя сообщества
 [ENTER для всех]:)
```

Отвечая на предыдущий вопрос, введите IP-адрес системы мониторинга сети. Если вы контролируете только свою машину, укажите IP-адрес обратной петли (127.0.0.1). Можно указать всю сеть в стандартном формате с косой чертой, представленном в главе 5 (например, 192.168.0.0/16). Наконец, при указании «0.0.0.0/0» любой пользователь Интернета сможет считать значения SNMP вашей системы, если получит (или разгадает) имя сообщества.

```
The OID that this community should be restricted to [RETURN for no-restriction]:
(OID, которым должно быть ограничено сообщество [ENTER при отсутствии ограничений]:)
```

Можно разрешить доступ различных сообществ SNMP только к отдельным частям дерева MIB. В данной книге не предусматривается обязательное выполнение таких действий, но вы можете ограничить доступ сообществ, выполнив более сложную настройку, нежели эта.

```
Finished Output: rocommunity  public 127.0.0.1       (Окончательный вывод)
Do another rocommunity line? (default = y):          (Еще одно сообщество?)
```

Если вы проводите мониторинг, описываемый в этой главе, вам нужно только одно сообщество SNMP. Ответьте n на предыдущий вопрос.

```
****************************************
*** Beginning trap destination setup ***                (Начало настройки ловушек)
****************************************
Do you want to configure where and if the agent will send traps? (default = y):
(Хотите ли вы конфигурировать расположение ловушек и сам факт их применения агентом?
(значение по умолчанию = y):)
```

Ловушка (*trap*) – это место, откуда система посылает уведомления другой системе об установке или сбросе того или иного MIB. Мы не будем их применять. Ответьте n.

```
****************************************
*** Beginning monitoring setup ***                    (Начало настройки мониторинга)
****************************************
Do you want to configure the agent's ability to monitor various aspects of your
system? (default = y):
(Хотите ли вы конфигурировать способность агента осуществлять мониторинг различных
аспектов вашей системы? (значение по умолчанию = y):)
```

Мы не применяем мониторинг процессов прямо сейчас, однако мы будем контролировать дисковое пространство. Ответьте y на предыдущий и последующий вопросы.

```
Do you want to configure the agents ability to monitor disk space? (default = y):
(Хотите ли вы конфигурировать способность агентов осуществлять мониторинг дискового
пространства? (значение по умолчанию = y):
Configuring: disk            (Конфигурирование: диск)
```

```
Description:                    (Описание:)
  Check for disk space usage of a partition.
  (Проверка потребления дискового пространства раздела.)
    The agent can check the amount of available disk space, and make
    sure it is above a set limit.
    (Агент может проверить объем доступного дискового пространства и убедиться,
     что данный объем выше установленного предела.)

      disk PATH [MIN=100000]

     PATH:  mount path to the disk in question.      (путь монтирования диска)
     MIN:   Disks with space below this value will have the Mib's errorFlag set.
            Can be a raw byte value or a percentage followed by the %
            symbol.  Default value = 100000.
    (MIN: для дисков, у которых объем доступного пространства меньше этого значения,
    будет установлен флаг ошибки MIB (Mib's errorFlag). Может быть указано в байтах
     или процентах, за которыми следует символ %. Значение по умолчанию = 100000.)

    The results are reported in the dskTable section of the UCD-SNMP-MIB tree
    (Результаты помещаются в раздел dskTable дерева UCD-SNMP-MIB)

Enter the mount point for the disk partion to be checked on: /usr
Enter the minimum amount of space that should be available on /usr: 10%
Finished Output: disk  /usr 10%
(Введите точку монтирования дискового раздела, подлежащего проверке: /usr
Введите минимальный объем пространства, который должен быть доступен в /usr: 10%
Окончательный вывод: disk  /usr 10%)
```

Введите здесь каждый раздел вашей системы. Информацию о разделах можно получить в */etc/fstab* или с помощью df(1). Вообще говоря, полезно предупреждать пользователей, когда на том или ином разделе останется меньше 10% свободного пространства.

```
Do you want to configure the agents ability to monitor load average? (default = y): n
Do you want to configure the agent's ability to monitor file sizes? (default = y): n
(Хотите ли вы конфигурировать способность агентов контролировать среднюю нагрузку?
(значение по умолчанию = y): n
Хотите ли вы конфигурировать способность агентов контролировать размеры файлов?
(значение по умолчанию = y): n
```

Мы не будем контролировать среднюю нагрузку и размеры файлов, поэтому на предыдущие два вопроса ответьте n. Эти параметры можно настроить позднее, когда вы больше узнаете об SNMP и мониторинге.

```
The following files were created:                    (Были созданы следующие файлы:)

  snmpd.conf installed in /usr/local/share/snmp     (snmpd.conf установлен в...)
#

       Stop and start snmpd to read the new configuration.
       (Остановите и запустите snmpd для считывания новой конфигурации.)

# killall snmpd && snmpd
#
```

Поздравляю! Теперь конфигурация SNMP полностью готова. Покиньте snmpwalk и посмотрите, какую информацию можно почерпнуть из вашей системы.

Индексные номера

Теперь рассмотрим параметры, которые обычно смущают новых пользователей SNMP. Возьмем следующий фрагмент вывода snmpwalk, описывающий диски системы. Помните: snmpwalk считывает информацию SNMP, доступную на сервере; это небольшой фрагмент вывода всей информации, которую можно получить через snmpd.

```
enterprises.ucdavis.dskTable.dskEntry.dskIndex.1 = 1
enterprises.ucdavis.dskTable.dskEntry.dskIndex.2 = 2
enterprises.ucdavis.dskTable.dskEntry.dskIndex.3 = 3
enterprises.ucdavis.dskTable.dskEntry.dskPath.1 = /usr
enterprises.ucdavis.dskTable.dskEntry.dskPath.2 = /
enterprises.ucdavis.dskTable.dskEntry.dskPath.3 = /var
enterprises.ucdavis.dskTable.dskEntry.dskDevice.1 = /dev/ad0s1f
enterprises.ucdavis.dskTable.dskEntry.dskDevice.2 = /dev/ad0s1a
enterprises.ucdavis.dskTable.dskEntry.dskDevice.3 = /dev/ad0s1e
```

Здесь приведены все разделы, представленные в *snmpd.conf*. В первых трех строках предыдущего примера мы видим, что `dksIndex` – это номера от 1 до 3. Применив snmptranslate, мы выясним, что это ссылочные номера дисковых разделов, которые мы контролируем, при этом каждому разделу назначен уникальный индексный номер.

В трех следующих строках (`dskPath`) мы отображаем индекс 1 в `/usr`, индекс 2 – в `/`, а индекс 3 – в `/var`.

Далее, в трех следующих строках (`dskDevice`) мы снова видим три записи. Как их сопоставить? Чтением в обратном порядке. Например, мы видим, что `dskDevice.1` – это `/dev/ad0s1f`. Мы знаем, что запись 1 означает */usr*. Это говорит нам, что */usr* находится на данном физическом устройстве.

Поскольку деревья MIB основаны на информации, которую необходимо получить, а не на устройствах, к которым нужен доступ, то информация о разделах представляется неупорядоченно. Читать ее намного труднее. Однако, проявив небольшое терпение, вы сможете в ней разобраться.

Вы увидите индексные номера в любых отчетах SNMP, которые состоят из нескольких частей. Информация о дисковых разделах размещается в начале, но можно найти индексы почти для всех компонентов. Поищите описание этих индексов, оно расположено в верхней части соответствующего раздела.

Долгосрочный мониторинг с помощью MRTG

Для долгосрочного мониторинга применим программу, которая выдает запросы SNMP через конкретные интервалы и записывает собранные ответы. Наиболее популярные программы такого рода – cricket и MRTG. Обе про-

граммы включены в коллекцию «портов» FreeBSD и легко устанавливаются в этой системе. Здесь будет обсуждаться MRTG (*/usr/ports/net/mrtg*).

MRTG, или Multi-Router Traffic Grapher, использует данные SNMP для автоматической генерации отчетов в виде веб-страницы с красиво оформленными графиками. MRTG можно запускать как демон, однако по традиции это задание cron, которое выполняется каждые пять минут. На машине, запускающей MRTG, необходим веб-сервер.

Программа MRTG может предоставлять руководителям, менеджерам и другим сотрудникам удобный доступ к данным о производительности, не разрешая им доступ к серверу. Кроме того, поскольку MRTG хранит записи на протяжении года, на их основании можно сделать выводы о наметившихся тенденциях. Также программа MRTG довольно полезна для выяснения затрат на аппаратное и программное обеспечение, поскольку позволяет точно сказать, в течение какого времени CPU занят и как изменилась производительность машины при добавлении той или иной программы.

Конфигурирование MRTG

С помощью прилагаемого инструмента cfgmaker(1) можно подготовить конфигурацию MRTG по умолчанию, которая оценивает сетевую пропускную способность интерфейсов:

```
# cfgmaker communityname@machine > mrtg.cfg
```

Например, если бы я запускал MRTG на моей локальной машине, используя имя сообщества «GetLostLoser», я выполнил бы такую команду:

```
# cfgmaker GetLostLoser@localhost > mrtg.cfg
```

cfgmaker выдает запросы SNMP указанному устройству и создает базовый файл настройки. В предыдущем примере вывод команды cfgmaker передается в файл *mrtg.cfg*. По умолчанию MRTG контролирует только сетевой трафик.

Прежде чем использовать конфигурационный файл, надо добавить в его начало директиву `WorkDir`. Эта директива предписывает программе MRTG, где хранить протоколы, графики, рабочие файлы и страницы HTML. Обычно я размещаю `WorkDir` в корневом каталоге веб-сервера:

```
WorkDir: /usr/local/share/apache/htdocs/mrtg
```

Вероятно, вам потребуется защитить этот каталог с помощью пароля, если веб-сервер общедоступен в Интернете или открыт миру иным образом (см. главу 15).

Пример записи mrtg.cfg

Сгенерированный *mrtg.cfg* содержит много информации, включая необязательный код HTML. Если изучить созданный файл, можно увидеть, что cfgmaker сгенерировал конфигурацию для каждого интерфейса. Это правильно, поскольку по умолчанию программа оценивает сетевую пропускную

способность. Интерфейс обратной петли и все неактивные интерфейсы закомментированы; оставшиеся незакомментированными строки представляют собой последовательность записей, подобную нижеследующей:

```
❶ Target ❷ [localhost.3]: 3:GetLostLoser@localhost
❸ MaxBytes[localhost.3]: 1250000
❹ Title[localhost.3]: petulance.blackhelicopters.org: xl0
❺ PageTop[localhost.3]: <H1>Traffic Analysis for xl0
 </H1>
 <TABLE>
   <TR><TD>System:</TD><TD>petulance.blackhelicopters.org in Basement Server Room</
TD></TR>
   <TR><TD>Maintainer:</TD><TD>mwlucas@blackhelicopters.org></TD></TR>
   <TR><TD>Interface:</TD><TD>xl0 (3)</TD></TR>
   <TR><TD>IP:</TD><TD>petulance.blackhelicopters.org  (192.168.1.100)</TD></TR>
   <TR><TD>Max Speed:</TD>
       <TD>1250.0 kBytes/s (ethernetCsmacd)</TD></TR>
  </TABLE>
```

Метки

Строка в квадратных скобках (❷) – это произвольная метка, указывающая на уникальную подсистему, находящуюся под контролем. В данном примере метка указывает на `localhost.3`. Это может быть сетевой интерфейс, дисковое пространство раздела, потребление процессора или что-либо другое. Имена всех файлов, созданных MRTG для данной подсистемы, начинаются с этой метки.

Для мониторинга любого количества компонентов вы должны назначить уникальную метку любой длины. Обычно я создаю метки, подобные `"webserver1.EthernetTraffic"` или `"webserver9.AvailableSwap"`. Целесообразнее присваивать меткам имена, подобные MIB, которые будут контролироваться. Однако имена MIB могут быть длинными, поэтому при назначении имен меток необходимо достичь приемлемого баланса.

Target

Ключевое слово `Target` (целевой объект, ❶) предписывает MRTG, какую машину опрашивать и для какого интерфейса данной машины предназначена эта конфигурация. (Действительный целевой объект следует за двоеточием.) Если вы изменяете имя сообщества или IP-адрес системы, эти параметры можно отредактировать прямо здесь. В данном случае целевой объект – `3:GetLostLoser@localhost`. В этой записи представлены номер интерфейса, сообщество SNMP и имя хоста.

Ранее, в разделе «Индексные номера», мы говорили об индексах в MIB. Если с помощью snmpwalk исследовать дерево MIB, то можно выяснить, что индексный номер этого интерфейса равен 3. Изучив вывод snmpwalk, можно выяснить, какому интерфейсу соответствует этот индексный номер. Однако каждый интерфейс имеет уникальный IP-адрес, по которому идентифици-

ровать интерфейс обычно проще. MRTG любезно приводит IP-адрес немного далее.

MaxBytes

MaxBytes (❸) – это максимальное значение, допустимое для данного компонента. Поскольку это сетевой интерфейс, то параметр MaxBytes равен максимальному количеству байтов, которое может обслуживать эта сетевая карта (карта 10BaseT). Программа MRTG достаточно разумна, чтобы выяснить значения MaxBytes для наиболее типичных сетевых устройств.

Вам никогда не придется изменять это значение при измерении пропускной способности. Значение MaxBytes надо менять при мониторинге других параметров, например потребления процессора и памяти.

Title и PageTop

Title (❹) и PageTop (❺) – это произвольный код HTML. Здесь можно размещать почти любые теги HTML, которые будут отображены на сгенерированной веб-странице MRTG.

Отредактированный файл настройки

Отредактированный мною базовый конфигурационный файл MRTG для мониторинга сетевого интерфейса выглядит так:

```
WorkDir: /usr/local/share/apache/htdocs/mrtg
Target[webserver1.EthernetTraffic]: 3:GetLostLoser@localhost
MaxBytes[webserver1.EthernetTraffic]: 1250000
Title[webserver1.EthernetTraffic]: Ethernet Interface
PageTop[webserver1.EthernetTraffic]: <H1>Traffic Throughput for Ethernet
Interface</H1>
<P>Call the Helpdesk if you have any questions
```

Сначала вы увидите, что метка изменена на нечто более значащее. Кроме того, код HTML в PageTop существенно сокращен, поскольку я хорошо знаю, что собой представляет система и кто ее поддерживает. Если эти страницы предназначены для руководства, то после PageTop я могу добавить пару строк HTML, которые описывают деятельность машины и порядок интерпретации данных.

В одном конфигурационном файле можно перечислить любое количество машин или интерфейсов. Задайте подходящие параметры для вашей системы.

Тестирование MRTG

Когда подготовите необходимую конфигурацию, проверьте ее, запустив MRTG вручную несколько раз:

```
# mrtg mrtg.cfg
```

Первые два раза MRTG выдаст предупреждение о том, что она не может найти файлы протоколов. При последующих запусках такого сообщения не бу-

дет. Такое предупреждение не свидетельствует о наличии проблем. Но если будет получено сообщение об ошибке, согласно которому MRTG не может достичь целевого объекта, значит, запись `Target` неверно сконфигурирована – имя сообщества, имя хоста или MIB указаны неправильно. Проверьте их, устраните неполадки и попробуйте снова. Когда программа MRTG начнет работать «молча», добавьте ее в таблицу cron пользователя root, задав интервал запуска равным пяти минутам. (Для запуска MRTG можно создать специального пользователя, однако в этом нет необходимости.)

Вывод MRTG будет отправлен в каталог, указанный директивой `WorkDir` конфигурационного файла. Если вы следовали предыдущему примеру, то по адресу *http://<hostname>/mrtg/webserver1.EthernetTraffic.html* найдете замечательный график сетевого трафика с момента запуска MRTG. На графике есть две отдельных линии: одна для входящего трафика, другая – для исходящего. MRTG проводит попарные измерения. Пример графика MRTG представлен на рис. 19.1.

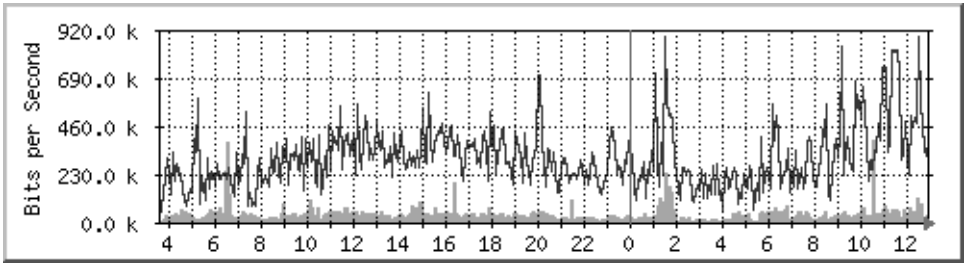

Рис. 19.1. *Простой график MRTG*

Измерьте с помощью MRTG трафик на одной из ваших машин, а затем приступим к измерению других интересных параметров, выбирая соответствующие SNMP MIB.

Отслеживание других системных значений

По умолчанию MRTG измеряет сетевой трафик, однако эта программа может получать любую информацию, доступную через SNMP. Достаточно указать MIB для информации, которую необходимо получить. Чтобы программа MRTG осуществляла мониторинг MIB, а не сетевого трафика, добавьте их в запись `Target`:

```
Target[label]:mibnumber1&mibnumber2:GetLostLoser@localhost
```

Скажем, для измерения системного времени процессора (MIB .1.3.6.1.4.1. 2021.11.9.0) и пользовательского времени процессора (MIB .1.3.6.1.4.1. 2021.11.10.0) подойдет такая строка:

```
Target[cpu]:1.3.6.1.4.1.2021.11.9.0&1.3.6.1.4.1.2021.11.10.0.:GetLostLoser@localhost
```

Напомним, MRTG строит графики MIB попарно, поэтому при мониторинге надо указывать соответствующие значения. Разумные варианты: «доступ-

ное и общее пространство свопа» или «системное и пользовательское время процессора». Измерение таких комбинаций как доступное пространство свопа и процентная доля свободного дискового пространства приведет к графикам, которые будет трудно понять.

Как узнать, какие MIB использовать? Изучим вывод snmpwalk, а затем преобразуем системное и пользовательское время процессора (соответственно `enterprises.ucdavis.systemStats.ssCpuUser.0` и `enterprises.ucdavis.systemStats.ssCpuSystem.0`). Мы увидим, что этим именам соответствуют номера `1.3.6.1.4.1.2021.11.9.0` и `.1.3.6.1.4.1.2021.11.10.0`. Чтобы вам не пришлось пробираться по дереву MIB в поисках необходимых данных, я облегчу вашу задачу.

Полезные MIB, относящиеся к net-snmp

Вот несколько MIB, относящихся к net-snmp, которые полезно контролировать. Они указаны по именам, а в круглых скобках представлены соответствующие числовые значения. Все эти MIB можно найти в `.1.3.6.1.4.1.2021`. Например, `Memory.MemTotalSwap` можно определить по запросу `1.3.6.1.4.1.2021.Memory.MemTotalSwap`.

Примечание

Перед использованием любого MIB обязательно протестируйте его с помощью snmpwalk. Каждый запрос должен возвращать одну строку. Если он возвращает несколько строк, то в конце вывода, вероятно, есть индекс, который надо указать перед вводом MIB в MRTG.

Без дальнейших предисловий рассмотрим некоторые полезные MIB:

`memory.memTotalSwap (4.3)`

Общее пространство свопа в системе. Проверьте это значение перед установкой пределов свопа в других графиках MRTG.

`memory.memAvailSwap (4.4)`

Объем оставшегося пространства свопа.

`memory.memTotalReal (4.5)`

Общая емкость памяти, действительно доступная в системе. Хотя вам не нужно осуществлять мониторинг этого значения (в конце концов, оперативная память в системе изменяется редко), его необходимо проверить перед установкой пределов в MRTG.

`memory.memAvailReal (4.6)`

Емкость неиспользованной памяти.

`memory.memBuffer (4.14)`

Емкость памяти в системном буфере.

`memory.memCached (4.15)`

Емкость памяти в системном кэше.

`dskTable.dskEntry.dskTotal (9.1.6)`

Общий размер раздела. Подобно другим общим значениям, этот размер – просто для справки. Данный MIB почти наверняка имеет индексы.

`dskTable.dskEntry.dskAvail (9.1.7)`

Объем пространства, оставшегося на диске, в блоках.

`dskTable.dskEntry.dskPercent (9.1.9)`

Процентная доля занятого дискового пространства.

`systemStats.ssCpuUser (11.9)`

Процентная доля времени процессора, которое затрачено программами, не связанными с ядром.

`systemStats.ssCpuSystem (11.10)`

Процентная доля времени процессора, потраченного на обслуживание ядра.

Есть масса других MIB, которые можно контролировать. Об SNMP и мониторинге написаны целые книги. Тем не менее перечисленные MIB охватывают базовые системные сервисы и предоставляют важную информацию для общих собраний.

Мониторинг отдельного MIB

Программа MRTG предназначена для попарного мониторинга параметров, но можно контролировать отдельный MIB, указав его дважды:

```
Target[localhost.memAvail]:.1.3.6.1.4.1.2021.4.6.0&.1.3.6.1.4.1.2021.4.6.0:GetLostL
oser@localhost
```

Таким образом, на графике MRTG можно представить только один MIB.

Примечание

Для целевого объекта обязательно укажите отдельную метку, которую следует применять для всех параметров настройки, относящихся к целевому объекту. В противном случае MRTG либо «пожалуется», либо перезапишет файлы протоколов других целевых объектов.

Настройка MRTG

Настроив работающую конфигурацию MRTG, будьте внимательны при тестировании новых конфигураций. Обычно новую конфигурацию MRTG я тестирую в отдельном каталоге WorkDir, поэтому ошибки в конфигурации не повредят существующие файлы протоколов или рабочие страницы состояния.

Файл *mrtg.cfg* содержит массу параметров, позволяющих настраивать почти каждый аспект оформления вывода и функциональности MRTG. Ниже представлены наиболее полезные параметры и описано, как применять их в рабочей среде.

`WithPeak[label]: wmy`

Протокол MRTG сводит старые записи к средним значениям, что приводит к уменьшению значений с течением времени. Параметр `WithPeak` заставляет MRTG хранить и отображать на графиках максимальные значения. В результате графики будут более сложными, однако они будут содержать больше полезной информации. Этот параметр можно установить для еженедельных (`w`), ежемесячных (`m`) и ежегодных (`y`) графиков либо комбинировать эти значения.

`MaxBytes[label]: number`

Это максимальное значение, которого может достичь MIB. (Метка (`label`) вводит в заблуждение, если MIB не измеряет количество байтов.) MRTG использует это значение, когда определяет, получен ли разумный ответ от устройства.

Значение `MaxBytes` задается для обоих измеряемых MIB, поэтому измеряйте разумные пары! Если вы проводите необычные измерения и вам нужны различные значения `MaxBytes`, применяйте `MaxBytes1` и `MaxBytes2`.

`YLegend[label]: text`

MRTG разместит этот текст вдоль графика. Поместите здесь описание измеряемого параметра, например `"% CPU Time"` (% времени процессора).

`LegendI[label] & LegendO[label]: text`

MRTG всегда измеряет два MIB. Первый MIB – традиционное значение `"In"`, а второй –`"Out"`. Здесь можно разместить краткие описания измеряемых параметров, которые будут располагаться под графиком.

`Legend1[label] & Legend2[label]: text`

`Legend1` – это метка для первого измеряемого MIB, а `Legend2` – для второго. Обе метки появятся в нижней части графика, в легенде.

`Legend3[label] & Legend4[label]: text`

Эти метки применяются при записи максимальных значений (с помощью параметра `WithPeak`). Если вы не записываете максимальные значения, эти метки не окажут никакого влияния.

`Directory[label]: directoryname`

Большие конфигурации MRTG могут приводить к созданию множества файлов. Ключевое слово `directory` позволяет размещать файлы с указанной меткой в подкаталоге WorkDir.

`Options[label]: option-names`

Параметры позволяют обслуживать специальные случаи. Все опции указываются в одной строке, после целевого объекта. Вот несколько хороших примеров:

`growright` – по умолчанию MRTG рисует графики справа налево. С помощью этой опции графики можно рисовать слева направо.

bits – на графиках будут отражаться не байты, а биты. Биты не только более выразительны, но и более точны. Точность зависит от измеряемого параметра. Например, многие администраторы измеряют пропускную способность в битах, а не в килобайтах.

gauge – как правило, SNMP сохраняет информацию в обратной форме. MRTG вычитает предыдущее значение из текущего, чтобы выявить изменение за последние пять минут. Параметр gauge можно использовать для SNMP MIB, которые не изменяются, например емкость диска.

unknaszero – когда по любой причине целевой объект недоступен (включая неполадки с питанием и проблемы с сетью), система берет для графика последнее известное значение. Насколько оно точное – вопрос спорный. Если задать этот параметр, то программа MRTG предполагает значение 0, если достичь целевого объекта невозможно.

Индексная страница MRTG

Последний этап настройки заключается в создании отдельной индексной страницы HTML для настройки MRTG. В состав MRTG входит инструмент для создания индексов, выполняющий эту операцию автоматически. Достаточно запустить такую команду:

```
# indexmaker mrtg.cfg > index.html
```

Возможно, вам потребуется отредактировать результаты, однако они являются отличной отправной точкой.

Образцы конфигураций MRTG

В этом разделе содержатся несколько образцов настройки MRTG. Чтобы задействовать их, вставьте правильные имена хоста и сообщества. Кроме того, обратите внимание, что операторы Target должны размещаться в одной строке; печатная полоса недостаточно широка, чтобы представить эту информацию кеглем шрифта, который можно прочесть без лупы. В каждом примере вам потребуется отредактировать оператор Target, указав надлежащие имя хоста и имя сообщества. Вероятно, вы отредактируете параметры PageTop, чтобы график отображал верную информацию о вашем хосте.

Пользовательское и системное потребление процессора

Первая конфигурация предназначена для сравнения пользовательского и системного потребления процессора в процентах. (Обратите внимание на параметр gauge.)

```
Title[myhost.AbsoluteBSD.com.cpu]: myhost.AbsoluteBSD.com CPU usage
YLegend[myhost.AbsoluteBSD.com.cpu]: CPU usage
WithPeak[myhost.AbsoluteBSD.com.cpu]: wmy
MaxBytes[myhost.AbsoluteBSD.com.cpu]: 100
Target[myhost.AbsoluteBSD.com.cpu]:.1.3.6.1.4.1.2021.11.9.0&.1.3.6.1.4.1.2021.11.10
.0:public@myhost.AbsoluteBSD.com
ShortLegend[myhost.AbsoluteBSD.com.cpu]: CPU
```

```
LegendI[myhost.AbsoluteBSD.com.cpu]: User CPU
LegendO[myhost.AbsoluteBSD.com.cpu]: System CPU
Legend1[myhost.AbsoluteBSD.com.cpu]: User Processor Usage
Legend2[myhost.AbsoluteBSD.com.cpu]: System Processor Usage
Legend3[myhost.AbsoluteBSD.com.cpu]: Maximal 5 Minute User Processor Usage
Legend4[myhost.AbsoluteBSD.com.cpu]: Maximal 5 Minute System Processor Usage
Options[myhost.AbsoluteBSD.com.cpu]: growright, gauge
PageTop[myhost.AbsoluteBSD.com.cpu]: <H1>CPU usage for myhost.AbsoluteBSD.com </H1>
```

Потребление свопа

Прежде чем измерять пространство свопа, необходимо узнать, каков объем свопа в системе. Эту информацию можно получить с помощью snmpwalk, запросив общий объем доступного пространства свопа, или выполнить `snmpwalk hostname community .1.3.6.1.4.1.2021.4.3`. Поместите это значение в поле `MaxBytes` и под `PageTop`, чтобы оно было показано на графике.

```
Title[myhost.AbsoluteBSD.com.swap]: myhost.AbsoluteBSD.com Available Swap
YLegend[myhost.AbsoluteBSD.com.swap]: swap
WithPeak[myhost.AbsoluteBSD.com.swap]: wmy
MaxBytes[myhost.AbsoluteBSD.com.swap]: 128000
Target[myhost.AbsoluteBSD.com.swap]:
.1.3.6.1.4.1.2021.4.4.0&.1.3.6.1.4.1.2021.4.4.0:public@myhost.AbsoluteBSD.com
ShortLegend[myhost.AbsoluteBSD.com.swap]: Swap
LegendI[myhost.AbsoluteBSD.com.swap]: Available Swap
LegendO[myhost.AbsoluteBSD.com.swap]: Available Swap
Legend1[myhost.AbsoluteBSD.com.swap]: Available Swap
Legend2[myhost.AbsoluteBSD.com.swap]: Available Swap
Legend3[myhost.AbsoluteBSD.com.swap]: Maximal 5 Minute Available Swap
Legend4[myhost.AbsoluteBSD.com.swap]: Maximal 5 Minute Available Swap
Options[myhost.AbsoluteBSD.com.swap]: growright, gauge
PageTop[myhost.AbsoluteBSD.com.swap]: <H1>Available Swap (out of 128M) for
myhost.AbsoluteBSD.com </H1>
```

Дисковый раздел

Измерить объем занятого пространства дискового раздела так же легко, однако необходимо знать индекс MIB для дискового раздела. (См. примеры в разделе «Индексные номера» ранее в этой главе.) Укажите правильный размер в поле `PageTop`.

```
Target[myhost.AbsoluteBSD.com.root]:.1.3.6.1.4.1.2021.9.1.9.1&.1.3.6.1.4.1.2021.9.1
.9.1:public@myhost.AbsoluteBSD.com
Title[myhost.AbsoluteBSD.com.root]:  Myhost.AbsoluteBSD.com Root partition % used
MaxBytes[myhost.AbsoluteBSD.com.root]: 100
WithPeak[myhost.AbsoluteBSD.com.root]: wmy
Suppress[myhost.AbsoluteBSD.com.root]: y
LegendI[myhost.AbsoluteBSD.com.root]:  % used
Legend1[myhost.AbsoluteBSD.com.root]:  % used
YLegend[myhost.AbsoluteBSD.com.root]: percent used
ShortLegend[myhost.AbsoluteBSD.com.root]: used
Options[myhost.AbsoluteBSD.com.root]: gauge, growright
PageTop[myhost.AbsoluteBSD.com.root]: <H1>myhost Root partition (% of 128M) used </H1>
```

Сетевой трафик

Для полноты представления рассмотрим образец конфигурации для измерения сетевого трафика. Предполагается наличие сетевого соединения 10 Мбит/с, а если вы применяете 100BaseT, то умножьте MaxBytes на 10.

```
Title[myhost.AbsoluteBSD.com.traffic]: myhost.AbsoluteBSD.com network traffic
MaxBytes[myhost.AbsoluteBSD.com.traffic]: 125000
WithPeak[myhost.AbsoluteBSD.com.traffic]: wmy
Target[myhost.AbsoluteBSD.com.traffic]: public@myhost.AbsoluteBSD.com
Options[myhost.AbsoluteBSD.com.traffic]: growright, bits
PageTop[myhost.AbsoluteBSD.com.traffic]: <H1>Network traffic on
myhost.AbsoluteBSD.com </H1>
```

Мониторинг систем не-BSD

MRTG может осуществлять мониторинг любых систем, которые применяют SNMP, так как это стандарт, поддерживаемый почти всеми поставщиками сетевого оборудования и операционных систем. Для проведения мониторинга необходимо получить список MIB, поддерживаемых продуктом, и знать их значение. Обычно поставщики предоставляют таблицы, истолковывающие MIB. Это простые текстовые файлы, объясняющие, как читать дерево MIB. Каждый файл начинается примерно с таких строк:

```
IPV6-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    mib-2, Counter32, Unsigned32, Integer32,
    Gauge32                              FROM SNMPv2-SMI
    DisplayString, PhysAddress, TruthValue, TimeStamp,
    VariablePointer, RowPointer          FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP,
    NOTIFICATION-GROUP                   FROM SNMPv2-CONF
    Ipv6IfIndex, Ipv6Address, Ipv6AddressPrefix,
    Ipv6AddressIfIdentifier,
    Ipv6IfIndexOrZero                    FROM IPV6-TC;
```

Скопируйте данные файлы описания в каталог вашей системы. Затем в переменной окружения MIBDIRS укажите этот каталог, после чего snmptranslate сможет распознать эти файлы.

SNMP можно настроить даже в системах NT; в результате их будет легче контролировать, а затраты составят лишь долю стоимости коммерческих систем. Единственное различие между MRTG и коммерческой системой заключается в том, что в случае MRTG вы должны отдавать себе отчет в своих действиях. Я настоятельно рекомендую сайт Гарта Уильямса (Garth Williams) «SNMP for the Public Community» (*http://www.wtcs.org/snmp4tpc/*), где рассматривается SNMP на других платформах. Заранее предупреждаю: реализация SNMP в продуктах Microsoft не предусматривает адекватного описания сообщений об ошибках. Windows 2000 и XP несколько лучше, чем NT.

20

Аварии и паника в системе

Одна из замечательных особенностей FreeBSD – стабильность этой системы; синий экран смерти появляется лишь при активизации заставки. На самом деле прошел почти год, прежде чем я осознал, что аварии на машине FreeBSD могут происходить не только из-за плохих аппаратных средств.

Во FreeBSD могут случаться аварии или *паника*, однако систему можно легко восстановить, поэтому впадать, извините, в панику не стоит. Даже когда система полностью блокирована и не отвечает на запросы, можно удаленно подключиться к консоли и инициировать перезагрузку. FreeBSD предоставляет инструменты, с помощью которых можно точно определить, что случилось. Система обеспечит вас подробной отладочной информацией. (Даже если непонятно, что делать с этой информацией, можно подготовить отчет о проблеме и обсудить этот вопрос с командой разработчиков FreeBSD.)

Что вызывает панику?

Когда система впадает в панику? Что ж, паника – это выбор, который делает ядро. Если система оказывается в ситуации, с которой не может справиться, либо внутренние проверки на непротиворечивость говорят о сбое, система сигнализирует о неполадках.

Если FreeBSD применяется в качестве настольной системы, она может впадать в панику при выполнении различных операций, большинство из которых связаны с аппаратным обеспечением. К примеру, на лэптопе со старой системой FreeBSD можно вызвать панику вытаскиванием карты PCMCIA, если предварительно не остановить ее обслуживание. (К счастью, сейчас такого нет.) Сегодня для приведения системы в состояние паники надо сильно постараться и выполнить недопустимые операции, войдя в систему с правами пользователя root (или запустив -current).

Намного труднее вызывать панику на сервере. Как правило, неполадки происходят при неверной конфигурации ядра, исчерпании системных ресурсов либо из-за не известной ранее ошибки FreeBSD. Устранить первые две причины довольно просто – это обсуждается в главах 4 и 18. Последний случай вызывает наибольшее беспокойство.

FreeBSD – сложная система. Ни ее аристократическое происхождение, ни процесс разработки с открытым кодом не могут защитить ее от ошибок. (Некоторые специалисты утверждают, что программного обеспечения без ошибок не существует.) К счастью, наследие системы и процесс разработки предоставляют информацию для самостоятельной отладки FreeBSD. Кроме того, существуют инструменты, способные собирать информацию для специалистов, занимающихся отладкой. Натолкнувшись на непонятный код ошибки, вы быстро поймете, что для кого-то он имеет тот или иной смысл.

Примечание

Если вы не программист, не беспокойтесь. Подготовив свою систему к устранению неполадок, вы поможете кому-либо решить эту проблему. Будем надеяться, что вам никогда не придется отлаживать свою систему. Однако если такое случится, она будет настроена должным образом!

Что представляет собой паника?

Впадая в панику, система останавливает выполнение всех программ. Вместо обычных сообщений на консоли появляется примерно такое:

```
Fatal trap 12: page fault while in kernel mode
fault virtual address   =       0x80c0153a
fault code / supervisor write, page not present
instruction pointer = 0x8:0xc015aa84
stack pointer =         0x10:0xc7377e7c
frame pointer =         0x10:0xc7377e80
code segment =          base 0x0, limit 0xfffff, type 0x1b
             =          DPL 0, pres 1, def32 1, gran 1
processor eflags =      interrupt enabled, resume, IOPL=0
current process =       5 (syncer)
interrupt mask =        bio
trap number =           12
panic: page fault
```

У малоопытного системного администратора от таких сообщений кровь может застыть в жилах. Однако волноваться рано. Иногда FreeBSD выдает в какой-то мере дружелюбные сообщения, которые описывают неполадки. Они являются хорошей отправной точкой для начала исследования или по крайней мере дают ключевые слова для поиска в Google. Иногда система в случае паники выдает конкретные инструкции о настройке параметров ядра для предотвращения повторных ошибок. Другие сообщения о неполадках, подобные вышеприведенному, намного более загадочны.

Единственное слово, которое в этом сообщении кажется смутно знакомым, находится в четвертой строке снизу. Здесь мы видим, что текущий процесс – это процесс с названием «syncer». Большинство людей не знают, что это такое, а многие из тех, кто имеет о нем понятие, осторожны в своих действиях. «Загадочная паника» – это одна из худших ситуаций, возможных во FreeBSD.

Ответные действия при панике

Если система впала в панику, прежде всего надо скопировать сообщения о неполадках. Поскольку в случае паники FreeBSD теряет работоспособность, стандартные команды UNIX не действуют – система не позволит войти в нее с помощью SSH и запустить даже такие простые команды, как script(1). Консоль может быть намертво заблокирована либо можно «застрять» в отладчике. В любом случае необходимо проанализировать сообщение об ошибке.

Впервые получив сообщение, подобное предыдущему, я стал искать бумагу и карандаш. Нашел старый конверт и огрызок карандаша и пополз между серверной стойкой и необработанной кирпичной стеной. Я удерживал равновесие, одной рукой не давая упасть шестидюймовому черно-белому монитору, а другой рукой прижимая конверт к стене. Очевидно, у меня появилась третья рука, с помощью которой я записал сообщение о неполадках на конверте, который каким-то образом оказался под рукой. Наконец, поцарапанный и согнутый в три погибели, я задним ходом выбрался из-за стойки и победоносно набрал и отправил это сообщение по электронной почте. Уж конечно, первоклассные разработчики FreeBSD посмотрят на весь этот мусор и точно скажут, что случилось.

Затратив такие усилия, я получил первый ответ, который обманул мои ожидания: «Можете прислать вывод обратной трассировки?»[1]

В почтовых рассылках FreeBSD я видел множество сообщений о подобных проблемах. Ответ всегда был тот же самый. Большинство отправителей таких сообщений больше не давали о себе знать. Я хорошо понимаю, что они чувствовали. Когда имеешь дело с неполадками на сервере или (еще хуже) с аварийным состоянием, меньше всего хочется заниматься реконфигурированием.

Неприятность с сообщением о неполадках, записанном на моем конверте, заключалась в том, что оно предоставляло лишь крохотный отрывок всей истории аварии. По сути, сообщение было настолько непонятным, что напоминало бессвязное описание угнанного автомобиля: «красный, с царапиной на крыле». Если вы не сообщите марку, модель автомобиля, номер VIN или номерной знак, то полиция продвинется недалеко. Точно так же без дополнительной информации от аварийного ядра разработчики FreeBSD не смогут выявить причины неполадок.

Однако есть простой путь решить эту проблему: подготовить сервер к возможной панике до ее возникновения. Проведите надлежащую настройку при установке сервера. Таким образом, при возникновении аварии вы автоматически получите отладочную информацию. Эта идея может показаться оригинальной, и ей, безусловно, не придается особое значение в документа-

---

[1] Обратная трассировка – это путь, по которому программа добралась до того места, в котором что-то произошло. Прослеживается по стеку вызовов и представляет собой обычно последовательность вызовов. – *Примеч. науч. ред.*

ции FreeBSD, однако имеет смысл подготовиться к аварии. Если она никогда не случится, что ж, не на что и жаловаться. Если же система впадет в панику, вы готовы предоставить разработчикам FreeBSD полный отладочный дамп через секунду после возникновения проблемы.

Необходимые условия

Чтобы подготовиться к неполадкам в ядре, надо установить исходный код системы. Кроме того, необходим один или несколько разделов свопа, размер которых по крайней мере на 1 Мбайт превосходит емкость физической памяти. Желательно, чтобы раздел свопа был в два раза больше оперативной памяти. Например, если емкость оперативной памяти составляет 512 Мбайт, то размер раздела свопа должен быть равен 513 Мбайт и более, а его предпочтительный размер – 1024 Мбайт. (Безусловно, на сервере должно быть несколько разделов свопа на разных дисках!) Если раздел свопа недостаточно велик, надо либо добавить жесткий диск с достаточным разделом свопа, либо переустановить систему. (Наличие в разделе */var* свободного пространства такого объема полезно, но необязательно.)

Если вы следовали указаниям по установке, приведенным в начале книги, то у вас все готово.

Получение аварийного дампа

Процесс получения аварийного дампа примерно таков. Если система сконфигурирована правильно, то при аварии она сохранит дамп оперативной памяти. Его нельзя сохранить в файле потому, что при аварии ядро ничего не знает о файлах, а знает только о разделах. Проще всего записать этот дамп в раздел свопа. Дамп размещается как можно ближе к концу этого раздела. После сохранения дампа памяти в свопе система перезагрузит компьютер.

Во время перезагрузки */etc/rc* активизирует раздел свопа. Затем (вероятно) запустит fsck на поврежденных дисках. Система должна включить свопинг до запуска fsck потому, что этой программе необходимо пространство свопа. Будем надеяться, что в системе достаточно места для свопинга, чтобы fsck могла получить все необходимое без перезаписи файла дампа, скрытого в разделе свопа.

Если в системе есть место, где может храниться дамп памяти, она проверяет раздел свопа на предмет наличия дампа. Если в разделе свопа найден дамп памяти, программа savecore копирует его в соответствующий файл, а затем удаляет дамп из свопа. После этого процесс перезагрузки продолжается. Теперь у вас есть файл дампа ядра, который можно применить для получения вывода обратной трассировки.

Отладка ядра

При стандартной установке ядра система FreeBSD удаляет всю отладочную информацию из ядра до его установки. В числе прочего удаляются *символы* (*symbols*), обеспечивающие отображение между машинным и исходным кодом.

Такие отображения могут быть больше самой программы, и никому не надо, чтобы ядро было в три раза больше действительно необходимого! Однако без этого отображения и другой отладочной информации не обойтись при выявлении ошибок, приведших к аварии.

Кроме того, это отображение содержит полный список номеров строк исходного кода. Следовательно, разработчик может точно выяснить, где произошли неполадки. Без этой информации разработчику пришлось бы соотносить дамп ядра и исходный код вручную. Примерно так собирают мозаику из миллиона кусочков, когда под рукой нет исходной картинки и когда неизвестно, все ли кусочки есть в наличии. В общем и целом это нелегкая задача. Она становится еще труднее, если разработчик, который должен предложить решение, является добровольцем. Вот почему отладочное ядро должно включать в себя символы.

Для сохранения символов добавьте в конфигурацию ядра следующие строки:

```
options         DDB
makeoptions     DEBUG=-g
```

Параметр `DDB` устанавливает отладчик ядра DDB. (Это не строго обязательно, однако такой отладчик может пригодиться. Он не занимает много места.) Параметр `makeoptions` предписывает системе собирать отладочное ядро.

Поведение системы после возникновения паники

При конфигурировании системы надо решить, как система должна работать после возникновения паники. Должен ли компьютер автоматически перезагрузиться, или экран с сообщением о панике должен сохраняться до ручной перезагрузки? Если дело происходит на удаленной системе, вам, безусловно, понадобится автоматическая перезагрузка. Однако если вы сидите за консолью и отлаживаете изменения ядра, то будете сами перезагружать систему.

Для автоматической перезагрузки надо включить параметр ядра `DDB_UNATTENDED`:

```
options DDB_UNATTENDED
```

Если этот параметр не включать, система будет ждать, пока вы не начнете перезагрузку.

kernel.debug

Сконфигурировав ядро по своему усмотрению, приступайте к его установке (см. главу 4).

После установки нового ядра в каталоге компиляции ядра появится файл *kernel.debug*. Это ваше ядро с символами – сохраните его где-нибудь. В следующий раз, когда вы будете обновлять систему или настраивать ядро, отладочное ядро может быть перезаписано новым отладочным ядром. Если ядро собрано просто для тестирования, убедитесь, что у вас есть отладочное ядро, не вызывающее никаких сомнений.

Одна из обычных причин неудачной отладки – потеря отладочного ядра и попытка отладить аварийное ядро с другим *kernel.debug*. Ничего хорошего из этого не получится. Как правило, я копирую *kernel.debug* в каталог */var/crash/kernel.debug.date*. Таким образом, я могу сказать, когда было собрано то или иное ядро. Благодаря этому я могу установить соответствие между текущим и отладочным ядром по дате их создания. Кроме того, я смогу определить, когда файл *kernel.debug* достаточно стар, чтобы его можно было удалить.

Если вам повезет, отладочные ядра вам никогда не понадобятся, но я на собственном опыте убедился, что лучше не уповать на везение. Отладочные ядра занимают на диске не много места и обеспечивают возможность быстро получить ответ при возникновении неполадок, поэтому я настоятельно рекомендую их использовать.

dumpon

Пора указать системе, куда записывать дамп памяти. Это место – *dumpdev*. FreeBSD применяет раздел свопа как устройство дампа (dump device), – вот почему этот раздел должен быть немного больше физической памяти. (Можно задействовать раздел UFS, однако после аварии он будет непригоден!)

Имя устройства можно получить из */etc/fstab*. Поищите строку с записью `swap` (в поле типа файловой системы – `FStyp`); первая запись в этой строке – имя физического устройства. Например, на моем лэптопе поле `swap` в */etc/fstab* выглядит так:

```
/dev/ad0s4b              none              swap     sw              0        0
```

Для работы с устройством дампа система должна применять dumpon(8), которая должна устанавливаться при каждой загрузке системы. Конечно, вы догадались, что в *rc.conf* есть соответствующий переключатель. Мой раздел свопа – */dev/ad0s4b*, поэтому в */etc/rc.conf* я укажу это устройство в качестве устройства дампа:

```
dumpdev="/dev/ad0s4b"
```

savecore

Далее следует указать системе, где хранить дамп после перезагрузки с помощью savecore(8). Значение по умолчанию, `/var/crash`, можно изменить с помощью параметра `dumpdir` в *rc.conf*. (Этот каталог должен существовать, так как savecore его не создаст!)

Когда вы приобретете опыт сохранения отладочной информации, вам может понадобиться регулирование этого процесса. Прочтите savecore(8) и установите соответствующие ключи в `savecore_flags` файла */etc/rc*. Часто применяется ключ `-z`, предписывающий сжатие файла дампа и позволяющий сберечь дисковое пространство. Программа savecore достаточно разумна и позволяет автоматически исключать из дампа неиспользуемую память.

После аварии

Если вы сидите перед компьютером, на котором произошла авария, то видите сообщение о неполадках. А если в системе задана автоматическая перезагрузка, то на экране появятся числа, сообщающие о количестве мегабайт памяти, которые будут подвергнуты дампу. Наконец, компьютер перезагрузится. Будет запущена программа fsck, а затем можно наблюдать, как savecore копирует дамп памяти из свопа в файл.

Если после аварии система не перезагружается автоматически, то в приглашении отладчика нужно ввести две команды: `panic` для синхронизации дисков и `continue` для продолжения процесса перезагрузки. (FreeBSD поддерживает многие варианты отладки, однако для их применения необходимо уметь работать с отладчиком ядра.)

Дампы и плохие ядра

Некоторые ядра погибают уже во время загрузки либо работают недостаточно долго для того, чтобы можно было устранить проблему. В этом случае надо загрузиться с другого ядра.

Сложность заключается в том, что программе savecore для создания образа дампа нужен файл ядра. По умолчанию savecore обращается к загрузочному ядру. Если после аварии система загружается с другого ядра, то программу savecore надо запустить вручную, чтобы указать, где находится правильный файл ядра. Прервите загрузку во время первоначального обратного отсчета и загрузите систему в однопользовательском режиме с помощью такой команды:

```
ok boot -s
```

Когда система выдаст приглашение на ввод команды, то первым делом проверьте систему с помощью fsck. После аварии диски почти всегда «грязные»:

```
# fsck -p
```

(В случае современных (огромных) дисков это может отнять несколько минут.)

Когда fsck завершит работу, смонтируйте файловую систему, в которой хранятся файлы ядра:

```
# mount /var
```

Наконец, сохраните дамп ядра, используя соответствующий файл ядра. Для этого программе savecore с помощью ключа `-N` надо указать, какой файл ядра следует задействовать. Если аварийное ядро – */kernel.bad*, выполните такую команду:

```
# savecore -N /kernel.bad /var/crash
```

Конечно, при выполнении дампа вручную можно задать дополнительные ключи savecore, например `-v` и `-z`.

Применение dump

Если вы разработчик ядра, то в этом месте вы перестанете меня слушать и будете полагаться на свой опыт отладки. Однако если вы начинающий системный администратор, то, вероятно, недостаточно искусны в языке C и не знаете внутренние параметры ядра, чтобы надеяться на его успешную отладку в сложных случаях. Поэтому сосредоточим внимание на извлечении информации, которая поможет разработчику идентифицировать проблему.

После проведения дампа в каталоге */var/crash* можно найти файлы *kernel.0* и *vmcore.0*. (Каждый последующий дамп получает следующий по порядку номер, например *kernel.1* и *vmcore.1*.) Файл *vmcore.0* – это действительный дамп памяти, а файл *kernel.0* – это копия ядра, приведшего к аварии. Для наших целей файл *kernel.0* бесполезен, но, на всякий случай, сохраните его. Файл *vmcore.0* жизненно важен.

При аварии можно скопировать отладочное ядро в */var/crash/kernel.debug.0*, чтобы хранить дампы в соответствии с их ядрами.

Примечание

Остальная часть процесса предоставляет отличную возможность для использования script(1).

Теперь запустим отладчик gdb. У gdb есть три аргумента. Аргумент `-k` предназначен для конфигурирования отладчика в соответствии с работой ядра. Два других аргумента – имя файла, содержащего ядро с символами, и имя дампа памяти:

```
# gdb -k kernel.debug.0 vmcore.0
```

После этого gdb выдаст информацию об авторских правах, сообщение о неполадках и копию дампа ядра. Пример сообщения о неполадках был приведен ранее, поэтому я не буду повторять его сейчас. Но мы еще не говорили о приглашении отладчика, которое вновь появляется после вывода всей этой информации:

```
(kgdb)
```

Теперь вы продвинулись дальше по сравнению с множеством людей, столкнувшихся с неполадками в системе. Похлопайте себя по плечу. Чтобы точно выяснить, где возникла паника, наберите `where` и нажмите Enter.

```
(kgdb) where
#0  dumpsys () at ../../../kern/kern_shutdown.c:505
#1  0xc0143119 in db_fncall (dummy1=0, dummy2=0, dummy3=0,
    dummy4=0xe0b749a4 " \ 0048\ 200%") at ../../../ddb/db_command.c:551
#2  0xc0142f33 in db_command (last_cmdp=0xc0313724, cmd_table=0xc0313544,
    aux_cmd_tablep=0xc030df2c, aux_cmd_tablep_end=0xc030df30)
    at ../../../ddb/db_command.c:348
#3  0xc0142fff in db_command_loop () at ../../../ddb/db_command.c:474
#4  0xc0145393 in db_trap (type=12, code=0) at ../../../ddb/db_trap.c:72
```

```
#5  0xc02ad0f6 in kdb_trap (type=12, code=0, regs=0xe0b74af4)
    at ../../../i386/i386/db_interface.c:161
#6  0xc02ba004 in trap_fatal (frame=0xe0b74af4, eva=40)
    at ../../../i386/i386/trap.c:846
#7  0xc02b9d71 in trap_pfault (frame=0xe0b74af4, usermode=0, eva=40)
    at ../../../i386/i386/trap.c:765
#8  0xc02b9907 in trap (frame={ tf_fs = 24, tf_es = 16, tf_ds = 16, tf_edi = 0,
      tf_esi = 0, tf_ebp = -524858548, tf_isp = -524858592,
      tf_ebx = -525288192, tf_edx = 0, tf_ecx = 1000000000, tf_eax = 0,
      tf_trapno = 12, tf_err = 0, tf_eip = -1071645917, tf_cs = 8,
      tf_eflags = 66182, tf_esp = -1070136512, tf_ss = 0} )
    at ../../../i386/i386/trap.c:433
#9  0xc01ffb23 in vcount (vp=0xe0b0bd00) at ../../../kern/vfs_subr.c:2301
#10 0xc01a5e58 in spec_close (ap=0xe0b74b94)
    at ../../../fs/specfs/spec_vnops.c:591
#11 0xc01a55f1 in spec_vnoperate (ap=0xe0b74b94)
    at ../../../fs/specfs/spec_vnops.c:121
#12 0xc0207454 in vn_close (vp=0xe0b0bd00, flags=3, cred=0xc32cce00,
    td=0xe0a8d360) at vnode_if.h:183
#13 0xc0207fab in vn_closefile (fp=0xc3369080, td=0xe0a8d360)
    at ../../../kern/vfs_vnops.c:757
#14 0xc01b1d50 in fdrop_locked (fp=0xc3369080, td=0xe0a8d360)
    at ../../../sys/file.h:230
#15 0xc01b155a in fdrop (fp=0xc3369080, td=0xe0a8d360)
    at ../../../kern/kern_descrip.c:1538
#16 0xc01b152d in closef (fp=0xc3369080, td=0xe0a8d360)
    at ../../../kern/kern_descrip.c:1524
#17 0xc01b114e in fdfree (td=0xe0a8d360) at ../../../kern/kern_descrip.c:1345
#18 0xc01b5173 in exit1 (td=0xe0a8d360, rv=256)
    at ../../../kern/kern_exit.c:199
#19 0xc01b4ec2 in sys_exit (td=0xe0a8d360, uap=0xe0b74d20)
    at ../../../kern/kern_exit.c:109
#20 0xc02ba2b7 in syscall (frame={ tf_fs = 47, tf_es = 47, tf_ds = 47,
      tf_edi = 135227560, tf_esi = 0, tf_ebp = -1077941020,
      tf_isp = -524857996, tf_ebx = -1, tf_edx = 135044144,
      tf_ecx = -1077942116, tf_eax = 1, tf_trapno = 12, tf_err = 2,
      tf_eip = 134865696, tf_cs = 31, tf_eflags = 663, tf_esp = -1077941064,
      tf_ss = 47} ) at ../../../i386/i386/trap.c:1049
#21 0xc02ae06d in syscall_with_err_pushed ()
#22 0x80503a5 in ?? ()
#23 0x807024a in ?? ()
#24 0xbfbfffb4 in ?? ()
#25 0x807daaf in ?? ()
#26 0x807d6eb in ?? ()
#27 0x80630c1 in ?? ()
#28 0x8062fed in ?? ()
#29 0x805ea4c in ?? ()
#30 0x8065949 in ?? ()
#31 0x806544d in ?? ()
#32 0x806dc17 in ?? ()
#33 0x80616b7 in ?? ()
```

```
#34 0x80613f0 in ?? ()
#35 0x8048135 in ?? ()
(kgdb)
```

Стоп! Данных довольно много. Если эту информацию и вывод `uname -a` послать по адресу *hackers@FreeBSD.org*, то разработчики обратят внимание на ваше письмо и окажут помощь. Возможно, в своем ответе они попросят набрать другие команды в приглашении `kgdb`, но в любом случае вы определенно завоюете их внимание. Вы встанете на путь устранения неполадок, а разработчики FreeBSD помогут справиться с ошибкой.

Расширенная отладка ядра

Никто не обвинит того, кто не знаком с программированием, если он здесь остановится. Но мы должны идти дальше. Поэтому без дальнейших предисловий посмотрим, что можно извлечь из отладочных сообщений, и попробуем выяснить, что следует включить в первое письмо о помощи. Не имея опыта работы с ядром, вы не сможете сами устранить неполадки, но сможете сузить проблему.

Собирая информацию до отправки письма о помощи, вы предотвращаете один или два цикла уточняющих вопросов в свой адрес. (Если вы уже просили о помощи в чрезвычайных обстоятельствах, то знаете, насколько это важно!) Не будучи хакером ядер, нельзя знать, какая часть информации наиболее важна, поэтому необходимо включить в письмо всю информацию, почерпнутую из вывода.

Прежде всего, следует осознать, что вывод обратной трассировки содержит команды, выполняемые ядром. Команды представлены в обратном порядке. Строка #1 описывает последнюю операцию, которую выполняло ядро перед дампом системы, представленным в строке 0. (Когда кто-либо говорит «перед» или «после», речь почти всегда идет о хронологическом порядке, а не о порядке операций, представленных в отладчике.)

В случае паники ядро вызовет либо функцию *trap*, либо (при наличии `INVARIANTS` в ядре) функцию *panic*. Существуют варианты trap и panic, например db_trap, однако вам нужны старые добрые trap или panic, без всяких украшений. Поищите эти функции в выводе gdb. В предыдущем примере функция trap есть в строке #8. Другие типы trap есть в строках 4–7, однако это не простые операторы trap, а вспомогательные функции, которые trap вызывает для точного выяснения произошедшего и выбора последующих действий.

Какая-то операция до строки #8 привела к панике. В строке #9 мы видим:

```
#9  0xc01ffb23 in vcount (vp=0xe0b0bd00) at ../../../kern/vfs_subr.c:2301
```

Шестнадцатеричные числа говорят не так уж много, но в этой строке, предшествующей панике, мы видим функцию `vcount`. Если запустить `man vcount`, то можно узнать, что vcount(9) – это стандартный системный вызов. Паника произошла при выполнении компилированного кода, начинающегося со строки 2301 файла */usr/src/sys/kern/vfs_subr.c*. (Все пути в дампах должны

начинаться в каталоге исходного кода ядра. Обычно это */usr/src/sys*.) Эта информация подсказывает разработчику, где искать причины неполадок.

Изучение строк

Рассмотрим строку #9 более подробно. Используйте команду up и номер строки, к которой вы хотите перейти:

```
 (kgdb) up 9
#9  0xc01ffb23 in vcount (vp=0xe0b0bd00) at ../../../kern/vfs_subr.c:2301
2301            SLIST_FOREACH(vq, &vp->v_rdev->si_hlist, v_specnext)
(kgdb)
```

Здесь мы видим действительную строку из файла *vfs_subr.c*. Код, полученный при компилировании этого файла, вызвал панику в системе. Вам не надо знать, что означает SLIST_FOREACH (между прочим, это макрос). Будет полезно углубиться дальше, однако из этого дампа можно выжать ненамного больше, если вы не знаете точно, как работает ядро.

Изучение переменных

Те, у кого есть минимальный опыт программирования, могут предположить, что элементы в круглых скобках после SLIST_FOREACH представляют собой переменные. Так оно и есть. Для каждой из этих переменных есть диапазон приемлемых значений. Специалист, который знает этот код, распознает такие значения. С помощью вывода содержимого каждой переменной можно перейти к процессу отладки. (Для вывода содержимого переменных в gdb надо задать команду p, указав имя переменной в качестве аргумента.)

Рассмотрим среднюю переменную vp:

```
(kgdb) p vp
$2 = (struct vnode *) 0xe0b0bd00
(kgdb)
```

Выражение (struct vnode *) сообщает нам, что это указатель на структуру данных. Содержимое структуры можно посмотреть, указав звездочку перед именем переменной:

```
(kgdb) p *vp
$3 = { v_flag = 8, v_usecount = 2, v_writecount = 1, v_holdcnt = 0,
  v_id = 6985, v_mount = 0x0, v_op = 0xc2d52a00, v_freelist = { tqe_next = 0x0,
    tqe_prev = 0xe083de1c} , v_nmntvnodes = { tqe_next = 0xe0b0b700,
    tqe_prev = 0xe0b0c024} , v_cleanblkhd = { tqh_first = 0x0,
    tqh_last = 0xe0b0bd2c} , v_dirtyblkhd = { tqh_first = 0x0,
    tqh_last = 0xe0b0bd34} , v_synclist = { le_next = 0x0, le_prev = 0x0} ,
  v_numoutput = 0, v_type = VBAD, v_un = { vu_mountedhere = 0x0,
    vu_socket = 0x0, vu_spec = { vu_specinfo = 0x0, vu_specnext = {
        sle_next = 0x0} } , vu_fifoinfo = 0x0} , v_lastw = 0, v_cstart = 0,
  v_lasta = 0, v_clen = 0, v_object = 0x0, v_interlock = { mtx_object = {
      lo_class = 0xc0335c60, lo_name = 0xc02ef5c1 "vnode interlock",
      lo_flags = 196608, lo_list = { stqe_next = 0x0} , lo_witness = 0x0} ,
    mtx_lock = 4, mtx_recurse = 0, mtx_blocked = { tqh_first = 0x0,
```

```
      tqh_last = 0xe0b0bd84} , mtx_contested = { le_next = 0x0, le_prev = 0x0} ,
    tsp = { tv_sec = 3584, tv_nsec = 101067509} ,
    file = 0xc02ef50a "../../../kern/vfs_subr.c", line = 1726,
    has_trace_time = 0} , v_lock = { lk_interlock = 0xc036e320,
    lk_flags = 16777216, lk_sharecount = 0, lk_waitcount = 0,
    lk_exclusivecount = 0, lk_prio = 80, lk_wmesg = 0xc02ef5d1 "vnlock",
    lk_timo = 6, lk_lockholder = -1} , v_vnlock = 0x0, v_tag = VT_NON,
  v_data = 0x0, v_cache_src = { lh_first = 0x0} , v_cache_dst = {
    tqh_first = 0x0, tqh_last = 0xe0b0bdd8} , v_dd = 0xe0b0bd00, v_ddid = 0,
  v_pollinfo = 0x0, v_vxproc = 0x0}
(kgdb)
```

Примечание

Для изучающих C это замечательный пример, показывающий, что работать с указателем проще, чем с объектом, на который он указывает.

Заинтересовавшийся разработчик может углубиться в рассмотрение этого вывода и разобраться, что происходит. Рассмотрим первую переменную `vq` и попробуем извлечь из нее подобную информацию:

```
(kgdb) p vq
$4 = (struct vnode *) 0x0
(kgdb)
```

Это не проблема, но мы застряли. Указатель, равный `0x0`, есть нулевой указатель. Он нужен по многим причинам, однако увидеть в нем мы ничего не можем. Тем не менее вы можете попробовать; дальнейшее изучение дампа с помощью gdb ничему не повредит.

```
(kgdb) p *vq
Cannot access memory at address 0x0.(Нет доступа к памяти по адресу 0x0)
(kgdb)
```

Возможно, вы никогда не слышали слова «нулевой указатель» (null pointer) в сочетании со словом «паника». Без глубокого рассмотрения кода ядра нельзя сказать, что панику вызвал именно этот указатель. В данном случае нулевой указатель совершенно правильный, а ядро впало в панику, когда пыталось решить, какое значение присвоить вновь выделенному указателю.[1]

Мнимая странность gdb

Можно еще больше углубиться в данные и увидеть, что случилось. Вторая переменная нашего вывода (`vp->v_rdev->si_hlist`) позволяет продвинуться немного дальше:

```
(kgdb) p vp->v_rdev
There is no member named v_rdev.          (Не существует элемента с именем v_rdev)
(kgdb)
```

[1] Как я это узнал? Я обменялся несколькими письмами с разработчиком ядра, обсудив этот дамп. Вот так.

Обычно это выражение помогает. У тех, кто уже работал с gdb, может возникнуть мысль, что программа ошибается. Однако в данном случае все верно. Здесь v_rdev – макрос для удобства, хотя это знают лишь специалисты, изучавшие код ядра. На самом деле v_rdev расширяется в v_un.vu_spec.vu_specinfo. От вас не требуется знать об этом, однако не удивляйтесь, если разработчик попросит вас набрать выражение, которое отличается от представленного в выводе трассировки.

Для просмотра vp->v_rdev введите такую команду:

```
(kgdb) p vp->v_un.vu_spec.vu_specinfo
$5 = (struct specinfo *) 0x0
(kgdb)
```

Если вы дошли до этого места, то должны увидеть здесь нулевой указатель.

Результаты

В данном случае дополнительные «раскопки» дают разработчику быстрый ответ. Фрагмент содержимого структуры vp сразу же указывает на проблему.

```
v_type = VBAD
```

Это виртуальный дескриптор (vnode), который сейчас не используется и даже не должен быть в этой части системы. Разработчик перейдет непосредственно к этому фрагменту и попытается выяснить, почему система пытается присвоить новому vnode фиктивное значение.

Я получил этот дамп от разработчика ядра. Он прокомментировал его так: «Я пытался добиться того, чтобы vcount() возвращал 0 для неверных vnodes. Строго говоря, это не было неправильно, однако *действительная* ошибка где-то в другом месте, а ‘исправление’ vcount() просто ее прячет.» Это правильная позиция при возникновении подобных проблем – пользователи BSD хотят находить ошибки, а не затушевывать их. Однако в этом случае разработчик может попросить вас набрать в gdb другие команды, чтобы получить дополнительную информацию. Он даже может попросить вас послать файлы *kernel.debug* и *vmcore*.

vmcore и безопасность

В файле *vmcore* присутствует все содержимое системной памяти на момент возникновения паники. Сюда может входить различная информация, связанная с безопасностью. Злоумышленник может воспользоваться этими данными, чтобы проникнуть в вашу систему. В своем письме разработчик может попросить у вас копию этого файла. Для этого у него могут быть разумные основания: отладка пройдет легче, а количество писем с уточняющими вопросами значительно сократится. Тем не менее внимательно продумайте возможные последствия предоставления такой информации. Если вы не знаете специалиста, которому задаете вопрос, или не доверяете ему, то не следует посылать этот файл!

Однако если паника воспроизводима, можно произвести «холодную» перезагрузку системы, войти в однопользовательский режим и немедленно инициировать панику. Тогда если в системе не запущена ни одна программа, содержащая конфиденциальную информацию, и никто не набирает пароли в системе, то в дампе не окажется опасной информации. Таким образом, воспроизведение паники в однопользовательском режиме позволит сгенерировать «чистый» файл дампа памяти.

Для подготовки чистого файла дампа ядра введите `boot -s` в приглашении загрузчика. Система выдаст приглашение на ввод команд, после чего надо выполнить минимальную настройку для подготовки дампа и вызвать в системе панику:

```
# dumpon /dev/ad0s4b
# mount -art ufs
# /usr/local/bin/command_that_panics_the_system
```

Первая строка указывает системе, где разместить дамп (задайте здесь правильное имя раздела свопа). Вторая строка монтирует файловые системы «только для чтения», поэтому после паники не понадобится запускать fsck. (Поскольку вы знаете о приближении аварии, то зачем с помощью fsck проверять самого себя?) Наконец, вы запускаете команду, которая вызывает панику в системе. В зависимости от локальных настроек вам могут понадобиться дополнительные команды, однако вышеприведенные команды подойдут в большинстве случаев.

Нужны ли символы?

В заключение рассмотрим сессию отладки для той же самой паники и того же ядра, но без отладочных символов. Сравним этот вывод с тем, что был представлен ранее в этой главе.

```
(kgdb) where
#0  0xc01c5982 in dumpsys ()
#1  0xc0143119 in db_fncall ()
#2  0xc0142f33 in db_command ()
#3  0xc0142fff in db_command_loop ()
#4  0xc0145393 in db_trap ()
#5  0xc02ad0f6 in kdb_trap ()
#6  0xc02ba004 in trap_fatal ()
#7  0xc02b9d71 in trap_pfault ()
#8  0xc02b9907 in trap ()
#9  0xc01ffb23 in vcount ()
#10 0xc01a5e58 in spec_close ()
#11 0xc01a55f1 in spec_vnoperate ()
#12 0xc0207454 in vn_close ()
#13 0xc0207fab in vn_closefile ()
#14 0xc01b1d50 in fdrop_locked ()
#15 0xc01b155a in fdrop ()
#16 0xc01b152d in closef ()
#17 0xc01b114e in fdfree ()
```

```
#18 0xc01b5173 in exit1 ()
#19 0xc01b4ec2 in sys_exit ()
#20 0xc02ba2b7 in syscall ()
#21 0xc02ae06d in syscall_with_err_pushed ()
#22 0x80503a5 in ?? ()
#23 0x807024a in ?? ()
#24 0xbfbfffb4 in ?? ()
#25 0x807daaf in ?? ()
#26 0x807d6eb in ?? ()
#27 0x80630c1 in ?? ()
#28 0x8062fed in ?? ()
#29 0x805ea4c in ?? ()
#30 0x8065949 in ?? ()
#31 0x806544d in ?? ()
#32 0x806dc17 in ?? ()
#33 0x80616b7 in ?? ()
#34 0x80613f0 in ?? ()
#35 0x8048135 in ?? ()
```

Вот так. Никаких подсказок о том, где произошла паника, – только имена вызываемых функций. Возможно, очень опытному хакеру удастся точно определить в ядре фрагмент когда, где происходят эти системные вызовы. Если разработчик ядра действительно заинтересован в решении проблемы, он может получить такую информацию:

```
(kgdb) p vcount
$1 = { <text variable, no debug info>}  0xc01ffb00 <vcount>
(kgdb) up 9
#9  0xc01ffb23 in vcount ()
(kgdb) p/x 0xc01ffb23 - 0xc01ffb00
$2 = 0x23
(kgdb)
```

Команда `p/x` означает «печать в шестнадцатеричном виде». Здесь мы примерно выяснили, насколько глубоко в vcount() таится проблема. Если разработчик имеет подобное ядро, собранное с подобного исходного кода, он может выполнить следующее:

```
(kgdb) l *(vcount + 0x23)
0xc01fb913 is in vcount (../../../kern/vfs_subr.c:2301).
2296            struct vnode *vq;
2297            int count;
2298
2299            count = 0;
2300            mtx_lock(&spechash_mtx);
2301            SLIST_FOREACH(vq, &vp->v_rdev->si_hlist, v_specnext)
2302                    count += vq->v_usecount;
2303            mtx_unlock(&spechash_mtx);
2304            return (count);
2305    }
(kgdb)
```

Вот так. Нет способа получить информацию о плохом vnode. Разработчику остается изучить код и попытаться выяснить проблему (при условии чрезвычайной целеустремленности). В любом случае маловероятно, что разработчик, стремящийся помочь вам, имеет на своей машине такую конфигурацию, как ваша. Многие разработчики будут счастливы настроить такую систему в обмен на щедрую сумму в устойчивой валюте, но трудно ожидать, что это будет сделано бесплатно.

Последовательные консоли

Возможность собирать отладочную информацию сама по себе прекрасна, но что если это надо делать удаленно? Что если надо перезагрузить удаленную машину, когда она не отвечает на запросы по сети? Лучше всего это сделать с помощью последовательной консоли. Если бы у меня была последовательная консоль на моей первой системе, впавшей в панику, то мне не пришлось бы жонглировать ручкой, бумагой и монитором.

Настоящие аппаратные средства UNIX (такие как Alpha и SPARC) имеют аппаратные системные консоли. В этих системах можно подключить последовательный кабель к порту последовательной консоли и получить свободный доступ к BIOS, загрузочным сообщениям и параметрам запуска. (Большинство аппаратных средств x86 не предоставляют такой возможности, – чтобы зайти в BIOS или нажать пробел для прерывания загрузки, необходимо сидеть за консолью. Такой функциональностью обладает лишь малая часть системных плат Intel, например L440GX, однако это специальная функция, которую не сразу найдешь.)

Управление системой FreeBSD, расположенной на другом конце штата, может вызвать трудности. У вас есть два выбора: аппаратная последовательная консоль или программная последовательная консоль, встроенная во FreeBSD. Для любой консоли необходимо доступное последовательное устройство. Если две машины FreeBSD расположены рядом, их можно подключить друг к другу. В компаниях с множеством систем FreeBSD (таких как Lucent) применяются «терминальные серверы», единственная функция которых состоит в обслуживании последовательных соединений.

Аппаратная последовательная консоль

Ни одна операционная система не предоставит вам доступ к PC-совместимым сообщениям BIOS через последовательный порт. Такие сообщения появляются до запуска операционной системы и обращения к жестким дискам.

Для преодоления этих трудностей есть аппаратные решения. Самое лучшее из тех, что я видел, это PC Weasel (*http://www.realweasel.com*). Это видеоплата с последовательным портом вместо видеопорта. Подсоединив последовательный кабель к Weasel, можно манипулировать BIOS удаленно, прерывать загрузку для перехода в однопользовательский режим и выполнять любые операции в системе, как будто вы находитесь за консолью. (Есть и еще производители подобных устройств, однако либо для них необходимо специ-

ализированное клиентское программное обеспечение, либо они стоят намного дороже.)

Программные последовательные консоли

В состав FreeBSD входит программная последовательная консоль. При загрузке система FreeBSD решает, где разместить консоль. По умолчанию это монитор и клавиатура. Однако с помощью несложных манипуляций можно настроить консоль на последовательном порту, который должен быть в системе. (Все чаще и чаще компьютеры поставляются без поддержки старых аппаратных средств. Это означает, что в них нет последовательных портов и гнезд ISA; возможно, придется купить карту PCI с последовательным портом.)

У этой последовательной консоли есть два недостатка: она не активизируется до запуска загрузчика FreeBSD и не позволяет увидеть сообщения BIOS (но возможность влиять на процесс загрузки у вас есть). Тем не менее она достаточно хороша для большинства случаев.

Физическая настройка программной последовательной консоли

Для использования последовательной консоли нужен нуль-модемный кабель (его можно приобрести в любом компьютерном магазине или в интернет-магазине – обратитесь к *pricewatch.com*). Выберите лучший[1] кабель из имеющихся; когда в чрезвычайных обстоятельствах вам понадобится последовательная консоль, помехи в линии передачи будут совершенно некстати.

Подключите один конец нуль-модемного кабеля к порту последовательной консоли на сервере FreeBSD – по умолчанию это первый последовательный порт (COM1 или sio0, в зависимости от того, к какой операционной системе вы привыкли). Это значение можно изменить с помощью перекомпиляции ядра, но обычно проще использовать значение по умолчанию.

Другой конец нуль-модемного кабеля подключите к открытому последовательному порту другой системы. Я рекомендую либо другую систему FreeBSD, либо иную систему UNIX, либо терминальный сервер. Можно задействовать систему Windows, однако она не предоставляет возможности удаленного управления. (Да, в системе Windows можно применять VNC или PC Anywhere, однако вы столкнетесь со сложной настройкой, в то время как простой машины FreeBSD будет достаточно.)

Если у вас есть две удаленные машины FreeBSD и вы хотите применять на них последовательные консоли, просто подключите консольный кабель ко второму последовательному порту на другом сервере. При наличии трех машин можно объединить их в цепочку. Комбинируя группы по две и три машины, вы сможете разместить последовательные консоли на любом количестве систем FreeBSD. (Я работал с 30–40 машинами FreeBSD в одной комна-

[1] А не самый дорогой. – *Примеч. науч. ред.*

те, где установка мониторов была бы просто непрактична, а последовательные консоли приносили большую пользу.)

Примечание

Если вам становится трудно выяснить, какая машина присоединена к какому порту, имеет смысл приобрести терминальный сервер.

Конфигурирование программной последовательной консоли

Если вы используете только последовательную консоль, предпишите системе задействовать именно ее. Для этого добавьте в */boot/loader.conf* следующую запись:

```
console="comconsole"
```

Для того чтобы переключиться обратно к видеоконсоли, удалите или закомментируйте эту строку. Кроме того, в */boot/loader.conf* можно задать такую строку:

```
console="vidconsole"
```

Изменение конфигурации

В серверной комнате вам может понадобиться переключаться от стандартной консоли к системной и обратно. Как правило, я управляю большим количеством машин FreeBSD через последовательную консоль. Если в той или иной машине возникают неполадки, я переключаюсь на реальную консоль; при этом я не хочу изменять */boot/loader.conf*, а хочу, чтобы это происходило «само собой». Во FreeBSD реализовать такую возможность просто.

Одна из первых операций, которые любая система x86 выполняет при загрузке, это проверка наличия клавиатуры. Систему FreeBSD можно настроить так, чтобы при обнаружении клавиатуры она использовала обычную консоль, а при отстутствии клавиатуры – последовательную консоль. (Это лучшее решение в случае негибкого оборудования x86.) Для этого надо отредактировать файл */boot.config*, разместив в нем следующее:

```
-h
```

Это аргумент, передаваемый системе при загрузке, – аналогично прерыванию загрузки и выполнению `boot -h`. (Конечно, для этого необходима подключенная клавиатура, поэтому целесообразность такой возможности представляется спорной!)

Если в */boot.config* задан ключ `-h`, а у вас нет клавиатуры, то система задействует последовательную консоль. Такая проверка примитивна, но она достаточно хороша, если вы знаете правила.

Применение последовательной консоли

Доступ к последовательной консоли можно получить различными способами, для каждого из которых необходим второй компьютер. Если на втором

компьютере запущена операционная система Microsoft, программа Hyperterm предоставит вам доступ к последовательным портам (установите параметры 9600 бод, 8 бит, без проверки четности и 1 стоп-бит). Для использования карманного компьютера в качестве переносного терминала (я часто применяю Handspring Visor с последовательным соединительным модулем, или «крэдлом»[1]) запустите одну из нескольких бесплатных терминальных программ vt100. Если на втором компьютере также запущена система FreeBSD, можно задействовать терминальную программу FreeBSD. Поскольку эта книга посвящена FreeBSD, здесь будет рассмотрено именно это решение.

FreeBSD обращается к линиям последовательной передачи данных с помощью программы tip(1), позволяющей подключаться к удаленной системе, ангалогично telnet. Для запуска tip наберите:

```
# tip portname
```

Имя порта (portname) – это сокращение для указния номера последовательного порта и скорости, на которой он работает. Файл */etc/remote* содержит список имен портов. Большинство записей в этом файле являются пережитком тех дней, когда протокол UUCP был основным протоколом передачи данных, а последовательные терминалы были нормой, а не исключением из правил. Однако в конце этого файла есть несколько таких записей:

```
# Hardwired line
cuaa0b|cua0b:dv=/dev/cuaa0:br#2400:pa=none:
cuaa0c|cua0c:dv=/dev/cuaa0:br#9600:pa=none:

# Finger friendly shortcuts
com1:dv=/dev/cuaa0:br#9600:pa=none:
com2:dv=/dev/cuaa1:br#9600:pa=none:
com3:dv=/dev/cuaa2:br#9600:pa=none:
com4:dv=/dev/cuaa3:br#9600:pa=none:
```

Опытные системные администраторы UNIX знакомы с cuaa0b и cuaa0c. (Записи «com» добавлены для удобства тех, кто вырос в среде Windows.) Предположим, что у вас есть две машины FreeBSD, подсоединенные друг к другу. С помощью нуль-модемного кабеля последовательный порт 1 одной машины соединен с последовательным портом 2 другой. Пусть вы хотите обратиться к локальному последовательному порту 2 для взаимодействия с последовательной консолью другой системы:

```
# tip com2
connected
```

Что бы вы ни набрали, ничего другого вы не увидите.

Если вы зарегистрируетесь на другой системе и перезагрузите ее, то в окне tip вдруг появится следующее:

---

[1] Handspring Visor – это карманный компьютер (КПК) типа Palm с PalmOS 3.x, «крэдл» – устройство (в даном случае с последовательным интерфейсом), в которое вставляется КПК для обмена данными с настольным компьютером. – *Примеч. науч. ред.*

```
boot() called on cpu#1
Waiting (max 60 seconds) for system process `bufdaemon' to stop...stopped
Waiting (max 60 seconds) for system process `syncer' to stop...stopped

syncing disks... 8 8 2 2
done
Uptime: 21m20s
Rebooting...
cpu_reset called on cpu#1
cpu_reset: Stopping other CPUs
cpu_reset: Restarting BSP
cpu_reset_proxy: Stopped CPU 1
```

Далее при загрузке системы наступит долгая пауза. Если вы находитесь рядом с системой, то сможете увидеть пробегающие стандартные сообщения BIOS. Наконец, вы увидите примерно следующее:

```
Console: serial port
BIOS drive A: is disk0
BIOS drive C: is disk1
BIOS 639kB/523200kB available memory

FreeBSD/i386 bootstrap loader, Revision 1.0
(mwlucas@magpire.blackhelicopters.org, Sat Jul  7 18:40:47 EDT 2001)
Loading /boot/defaults/loader.conf
/boot/kernel/kernel text=0x1d7b58 data=0x3b28c+0x5132c
syms=[0x4+0x30420+0x4+0x3a852]
/
Hit [Enter] to boot immediately, or any other key for command prompt.
Booting [/boot/kernel/kernel] in 10 seconds...
```

Для прерывания загрузки нажмите клавишу пробела.

Теперь вы как будто находитесь за клавиатурой. Неважно, что система расположена за 1000 миль от вас, вы можете изменить загрузочное ядро, выполнить загрузку с выдачей подробных сообщений, перейти в однопользовательский режим и вручную запустить fsck на жестком диске и т. д. Последовательная консоль не убережет вас от неполадок в BIOS или аппаратных средствах, однако с ее помощью легче диагностировать другие компоненты.

Для продолжения загрузки ядра наберите `boot`. В конечном итоге вы увидите следующее:

```
Additional TCP options:.
Starting background filesystem checks

Tue Jul 10 19:40:21 EDT 2001
```

Все выглядит почти так же, как в случае обычной консоли, но нет приглашения на вход в систему.

Последовательный вход в систему

По умолчанию последовательные порты не рассматриваются как устройства для входа в систему. Многие пользователи применяют последовательные

порты для последовательных мышей или модемов, а не для консольных соединений. Если двигать мышью в ответ на приглашение войти в систему, то в лучшем случае система будет сбита с толку. Последовательный порт можно применять как порт мыши, модемный порт или консольный порт, но не одновременно. Пользователи, желающие войти в систему по последовательной линии, должны знать, как активизировать терминал на этом порту.

Для активизации терминала следует первым делом обратиться к файлу */etc/ttys* и получить список всех терминалов и псевдотерминалов системы. Последовательный порт – это тоже терминал. На «терминальном» языке устройство sio0/com1 известно как ttyd0, что означает «tty dialup 0».[1] Найдите строку `ttyd0`:

```
ttyd0   "/usr/libexec/getty std.9600"   dialup  off secure
```

Четвертое поле в этой строке говорит, что терминал отключен. Измените «off» на «on», и при следующем перезапуске демона init или процесса с ID 1 система предоставит терминальный сервис. Для получения ID процесса init не надо запускать ps, этот процесс всегда имеет ID 1. Если init не запускается, то не запускается и система. Ее можно изящно перезапустить с помощью сигнала `-1`.

```
# kill -1 1
```

После этого вы внезапно увидите активность на последовательной консоли.

```
Tue Jul 10 19:50:02 EDT 2001
FreeBSD/i386 (magpire.blackhelicopters.org) (ttyd0)

login:
```

Вход в систему при чрезвычайных обстоятельствах

При неблагоприятных обстоятельствах система FreeBSD не сможет взаимодействовать с сетью. Возможно, вышла из строя сетевая карта или система впала в панику при запуске той или иной программы. В моей практике был случай, когда сервер на другом конце штата запаниковал при загрузке, поскольку программа базы данных вызывала аварию всей системы. Система перезагружалась, проверяла диски, запускала программы и снова оказывалась в аварийной ситуации. В сценарий запуска программы, вызывающей аварию, надо было добавить специальный ключ командной строки.

Однако проблема заключалась в том, что система не позволяла войти в нее по SSH, зарегистрироваться под именем root и остановить базу данных. У меня была последовательная консоль, но она не разрешала вход в систему. Я сидел перед последовательной консолью и наблюдал сообщение о неполадках, появляющееся снова и снова. В голове теснились мысли: «Если бы я смог войти в систему по последовательному порту, то внес бы исправления

[1] Любознательный читатель может заинтересоваться, что означает «tty». Это «teletype» (телетайп). Да, система UNIX так стара.

в сценарий запуска». Посмотрим, как это можно сделать. Возможно, такой вход в систему вам никогда не понадобится, однако знать об этом полезно.

Поскольку FreeBSD не ограничивает доступ к консоли до того, как система перейдет в многопользовательский режим, можно прервать процесс загрузки, загрузиться в однопользовательском режиме и внести изменения. Для этого откройте соединение с последовательной консолью, а затем попросите сотрудника, находящегося рядом с удаленной машиной, выключить и включить ее. (Это не полезно для вашего компьютера, однако в повторяющихся авариях тоже ничего хорошего!) Загрузитесь в однопользовательском режиме и выполните следующую команду:

```
# fsck -p
```

Она ликвидирует последствия ущерба, причиненного выключением и включением питания. Теперь запустите такую команду:

```
# mount -a
```

Сейчас вы получили доступ к локальным файловым системам. Отредактируйте файл */etc/ttys* и разрешите вход в систему по последовательному порту, как описано в предыдущем разделе. Кроме того, можно отредактировать сценарий запуска, вызывающий аварии в системе.

Для завершения начальной загрузки покиньте командную оболочку. После этого вы сможет войти в систему через последовательную консоль.

Выключение последовательной консоли

В качестве управляющего символа программа tip(1) использует тильду (~). Чтобы выключить последовательную консоль, можно в любой момент ввести последовательность выключения «тильда-точка»:

```
~.
```

Вы грациозно отключитесь, но не выйдете из системы (последовательный протокол очень небрежен). Если вернуться назад и снова набрать `tip com2`, то вы снова окажетесь в предыдущей сессии. Если кабель сервера 1 подсоединен к консольному порту сервера 2, это может выглядеть так:

```
server1#tip com1
connected

server2#
```

У вас это может не вызвать беспокойства, однако взлом сервера 1 подвергнет риску сервер 2. Возьмите в привычку выходить из системы, когда соединяетесь по последовательной линии!

Предоставление отчета о проблеме

Вы можете возразить, что эту тему следовало бы обсудить в главе 2 «Как получить помощь». Название «Problem Report» (PR, отчет о проблеме) произ-

водит впечатление, не правда ли? Однако для предоставления PR необходим минимальный уровень информации, собрать которую поможет эта глава.

Система выдачи PR

Для работы с PR система FreeBSD использует GNATS – популярную базу данных для отслеживания ошибок. Основная база GNATS системы FreeBSD доступна по адресу *http://www.FreeBSD.org/support.html*. Чтобы PR был введен в GNATS, он должен быть представлен в соответствующем формате одним из двух способов.

Прежде всего вы можете представить свой PR по Сети. Потребуется много операций вырезания и вставки, но это, конечно, возможно. Форма PR доступна по адресу *http://www.FreeBSD.org/send-pr.html*.

Однако если система FreeBSD подключена к Сети, то проще всего применить программу send-pr(1). Она генерирует шаблон, который необходимо заполнить соответствующей информацией, и форматирует сообщение специально для использования в GNATS. Заплатки, предложения и отчеты об ошибках, представленные с помощью send-pr, постоянно записываются.

Что есть в PR?

PR не предназначен для сообщений о таких проблемах, как «моя сетевая карта не работает». Необходимо по возможности устранить неполадки с помощью почтовой рассылки или ее архивов. Программа send-pr предназначена для заплаток и отладочной информации.

Хороший PR содержит достаточно информации для выявления проблемы, а может быть, и предлагаемое решение. Если у вас есть время, посмотрите несколько открытых PR – в них можно почерпнуть те или иные сведения. Когда я писал эти строки, существовало 1918 открытых PR. Многие из них содержат солидную отладочную информацию. Другие PR менее полезны.

В FAQ системы FreeBSD есть шутка Дэга-Эрлинга С. Сморграва (Dag-Erling C. Smшrgrav): «Сколько пользователей -current требуется для того, чтобы заменить электрическую лампочку»? Ответ: «Три пользователя должны представить PR об этой проблеме, причем один из PR отнесен к категории doc и содержит только слово "темно"». Помните об этом, когда заполняете PR; включите отладочный вывод, а еще лучше – заплатку для устранения проблемы. Перед тем как открыть PR, следует тщательно обдумать, с проблемой какого рода вы столкнулись. Если проблема сводится к диагнозу «темно», то надо копать немного глубже.

Применение send-pr

Команда send-pr открывает текстовый шаблон в редакторе, задаваемом переменной `$EDITOR`. Когда вы заполните шаблон, send-pr отправит его в GNATS. Предполагается, что в системе настроена электронная почта. Если это не так, то для предоставления PR задействуйте веб-интерфейс (*http://www.FreeBSD.org/send-pr.html*). Вот образец шаблона:

```
To: FreeBSD-gnats-submit@freebsd.org
From: Michael Lucas <mwlucas>
Reply-To: Michael Lucas <mwlucas>
Cc:
X-send-pr-version: 3.113
X-GNATS-Notify:

>Submitter-Id:    current-users
>Originator:      Michael Lucas
>Organization:    <organization of PR author (multiple lines)>
>Confidential:    no <FreeBSD PRs are public data>
>Synopsis:  <synopsis of the problem (one line)>
>Severity:  <[ non-critical | serious | critical ] (one line)>
>Priority:  <[ low | medium | high ] (one line)>
>Category:  <choose from the list of categories above (one line)>
>Class:          <[ sw-bug | doc-bug | change-request | update | maintainer-update
] (one line)>
>Release:   FreeBSD 5.0-CURRENT i386
>Environment:
System: FreeBSD pedicular.blackhelicopters.org 5.0-CURRENT FreeBSD 5.0-CURRENT #5:
Wed Apr 24 07:27:19 EDT 2002 mwlucas@pedicular.blackhelicopters.org:/shared/usr/
currentobj/usr/src/sys/BLEEDING i386
     <machine, os, target, libraries (multiple lines)>
>Description:
     <precise description of the problem (multiple lines)>
>How-To-Repeat:
     <code/input/activities to reproduce the problem (multiple lines)>
>Fix:
     <how to correct or work around the problem, if known (multiple lines)>
```

Заполнение формы

Неважно, какой метод выбран. Главное – заполнить форму. Разберем ее по строкам.

`To`, `Subject`, `Submitter-Id` (Кому, Тема, Кто представил)

Эти строки можно не трогать. База GNATS позаботится о них сама.

`From` (От кого)

Убедитесь, что строка `From` содержит действительный электронный адрес. Именно по этому адресу служба поддержки GNATS или разработчик попытаются связаться с вами.

`Originator` (Источник)

Это ваше имя. Обычно оно берется из системного окружения. Некоторые пользователи Интернета скрываются под псевдонимами, но лучше разместить здесь свое настоящее имя. Трудно подходить к серьезной проблеме с должным вниманием, если вы скрываетесь под именем «Doctor Web».

`Organization` (Организация)

Здесь надо либо указать название организации, либо оставить это поле пустым.

`Confidential` (Конфиденциально)

Значение по умолчанию – `"no"`. GNATS – это общедоступная база данных. Если вы помещаете в PR конфиденциальную информацию, вы делаете что-то не так. (Если вы считаете, что обнаружили ошибку, относящуюся к безопасности, напишите письмо по адресу *security-office@FreeBSD.org*. Однако не поступайте так только из-за того, что в отладочную информацию включен пароль root.)

`Synopsis` (Краткий обзор)

Не исключено, что это самая важная строка. Дайте краткое, однострочное описание проблемы, поскольку разработчики зачастую именно по этой строке определяют, каким PR уделить внимание. Краткий обзор «Моя система не работает!» заслужит немногословный комментарий о бесполезности этого PR, а сообщение «паника в ядре при большой нагрузке на процессор, отладочный дамп присоединен» имеет больше шансов привлечь внимание опытных специалистов. Если у вас есть заплатка для устранения проблемы, поместите в описании слово «PATCH» (заплатка). Такому PR почти гарантирован быстрый ответ.

`Severity` (Сложность)

Это поле предоставляет три варианта; выберите из них наиболее подходящий. Если вы заслужите репутацию пользователя, представляющего мелкие ошибки как «критические», довольно скоро ваши PR будут игнорироваться. (Здесь все построено на честности, а репутация значит намного больше, чем можно подумать.)

`Priority` (Приоритет)

В каком-то смысле данное поле названо неправильно. Проблема может быть высокоприоритетной для вас, но разработчики могут проигнорировать это поле. Тем не менее здесь можно указать то или иное значение. Хорошее краткое описание получит лучший отзыв, чем «высокий» приоритет, однако указание высокого приоритета поможет вам выпустить пар.

`Category` (Категория)

Это поле имеет несколько вариантов, многие из которых устарели или потеряли смысл. Например, если вы столкнулись с неполадками в чьем-либо коде, то размещение PR в категории `GNU` вызовет ответ «обратитесь к авторам». В случае неполадок с ядром укажите категорию «kern».

`Class` (Класс)

Это поле содержит общее описание PR. Большинство вариантов не требуют объяснений. Если вы столкнулись с аварийным отказом программы или системы, укажите класс `"sw-bug"`.

`Release` (Выпуск)

Тип вашей системы автоматически вводится в поле `Release`. Если вы заполняете PR не в системе, где проявляются неполадки, а в какой-то другой, то укажите верное значение.

`System` (Система)

В заключение разместите вывод `uname -a` в поле `System`. В это поле можно добавить дополнительную информацию, описывающую другие компоненты операционной среды. Например, если система представляет собой сильно нагруженный сервер новостей, упомяните это. Если у вас есть фрагмент конфигурационного файла, который порождает панику, разместите его здесь.

`Description` (Описание)

Поле `Description` — это текстовый раздел, где в свободной форме можно подробно описать проблему. Не разглагольствуйте и не говорите бессвязно, а просто опишите, что случилось. Представьте все имеющиеся сообщения об ошибках. Именно здесь надо разместить отладочный вывод. Если отладочного вывода при неполадках в ядре у вас нет, не посылайте PR. Кроме того, разместите в этом поле конфигурацию ядра и содержимое */var/run/dmesg.boot*.

`How-To-Repeat` (Как воспроизвести)

В этом поле разместите фрагмент кода, серию команд или текстовое описание того, как воспроизвести проблему. В некоторых PR это поле может быть очень коротким. Предложение «прочитайте FreeBSD-questions за неделю и посмотрите, как часто это встречается» вполне подходит для `How-To-Repeat`, если речь идет об изменениях в документации. Более специфичные проблемы требуют большего количества информации.

`Fix` (Решение)

Самая важна часть PR преставлена в поле `Fix`. Если у вас есть заплатка, решающая проблему, разместите ее здесь. Если вы знаете обходной путь, расскажите о нем. Дайте здесь любую информацию, способствующую решению проблемы. Иногда самое необычное предложение или условие является жизненно важным для нахождения решения.[1]

Хороший PR всегда содержит что-нибудь в поле `Fix`. Возможно, ваше решение не удастся реализовать, однако оно демонстрирует ваши мысли по поводу этой проблемы. Невероятный объем помощи, которую команда FreeBSD оказывает через почтовые рассылки и веб-сайты, иногда вводит спрашивающего в заблуждение, заставляя забыть, что когда он оказывается в тупике, то ответственность за принятое решение возлагается на него.

Когда вы сохраните текст и выйдете из редактора, send-pr спросит вас, хотите ли вы представить на рассмотрение свой PR. Если вы думаете, что в PR есть достаточно информации для устранения проблемы, скажите «yes». Ваша система отправит письмо с этим PR.

Независимо от того, какой метод избран для предоставления PR, вы получите письмо с подтверждением приема. В теме сообщения будет номер PR –

[1] Если вы используете веб-интерфейс, не помещайте заплатки в поле `Fix`. Веб-форма для преставления PR преобразует все знаки табуляции в пробелы и разрушит форматирование заплатки.

обычно что-нибудь вроде «kern/22459» и предоставленное вами краткое описание. Любое письмо, посланное по адресу *FreeBSD-gnats-submit@FreeBSD.org* с такой темой сообщения, будет присоединено к этому PR. Заплатки и отзывы можно представить с любого компьютера, на котором работает почтовая система.

Аналогично любой отзыв, полученный на вашу заплатку, будет храниться вместе с PR. Состояние PR можно проверить по адресу *http://www.FreeBSD.org/cgi/query-pr.cgi*.

Примечание

Теперь ваш вклад навеки запечатлен в системе FreeBSD. Нет никакой гарантии, что он будет принят или что проблема будет решена; просто ваш PR будет записан и всегда доступен.

Результаты PR

Правильно заполненный PR обычно быстро приводит к решению проблемы. На сегодня я представил 59 PR. По большинству из них были найдены решения, приняты меры, или они просто были приняты к сведению. Некоторые из них представляли собой описания тривиальных ошибок в документации, лежащей в */usr/src/contrib*. Здесь Проект намеренно снимает с себя ответственность за мелкие исправления. Более 90% моих PR привели к желаемому результату, и такого результата может добиться каждый.

Если вам довелось столкнуться с проблемами в той части системы, с которой мало кто знаком, какое-то время ваш PR будет чахнуть без внимания. Если вам кажется, что ваш PR забыт, пошлите дружелюбное сообщение в соответствующую почтовую рассылку, указав номер PR, а также кратко объяснив суть проблемы и почему она так важна. Поскольку FreeBSD поддерживается усилиями добровольцев, то вполне возможно, что со специалистом, который обрабатывает PR соответствующего типа, что-то случилось. Многие разработчики FreeBSD – профессиональные программисты, однако для многих из них FreeBSD – это хобби, которое отходит на задний план перед больными детьми или важными сроками завершения той или иной работы. Если ничего другого не остается, вы можете обратиться в компании, оказывающие платную поддержку. Их список представлен на веб-сайте FreeBSD.

Неожиданно много трудных PR быстро приводят к решению проблемы, если в них представлена надлежащая информация. Вспомните, что разработчики FreeBSD поддерживают систему из любви к ней, а не по принуждению. Они хотят производить качественный код. Это более сильная мотивация по сравнению с денежным чеком. Если вы помогаете им производить качественный продукт, они будут рады работать с вами.

Поздравляю! Теперь вы готовы к авариям, как любой разработчик, не имеющий отношения к созданию ядра. Надлежащая подготовка может сделать вашу жизнь легче. Готовность к худшему – это один из лучших способов обеспечить себе спокойный сон.

21

Настольная система FreeBSD

Зачем применять FreeBSD на настольных компьютерах? Ну а почему бы и нет? В качестве настольной системы ОС FreeBSD так же хороша, как и в качестве сервера. И хотя разработка FreeBSD в основном сосредоточена на серверной части, устойчивость к авариям и стабильность делают FreeBSD замечательной рабочей системой.

Настольная система FreeBSD предоставит вам доступ к локальной сети Windows NT. Вы сможете совместно использовать файлы, бродить по Сети, читать письма и составлять документы, и все это без единой аварии. Кроме того, применение FreeBSD в качестве настольной системы – это замечательный способ изучить UNIX и укрепить веру в себя как в системного администратора.

Примечание

Было время, когда термин «рабочая станция» означал «компьютер UNIX». Под системой UNIX понималась SunOS, прямая наследница BSD4.2. Если BSD так давно применяется в качестве рабочей станции, то она подойдет и для вас.

В этой главе не рассказывается о том, как превратить FreeBSD в удобную настольную систему, поскольку эта тема достойна отдельной книги. В следующем разделе будут подробно рассмотрены программные средства FreeBSD, обеспечивающие поддержку SMB (Server Message Block, SMB). Кроме того, будут указаны полезные программы, описание которых можно найти в другом месте. Почерпнув сведения о FreeBSD из данной книги и обратившись к архивам почтовых рассылок, вы сможете работать с этими инструментами.

Доступ к разделяемым файловым ресурсам

В типичной офисной сети стандартным сетевым протоколом совместного использования файлов является общий протокол доступа к файлам Интернета (Common Internet File Sharing, CIFS) от Microsoft. (Одно время CIFS назывался Server Message Block (SMB, блок серверных сообщений.) Это типичное «сетевое окружение» (Network Neighborhood), к которому имеют доступ

пользователи Windows. Первоначально этот протокол предоставлялся только операционными системами Microsoft Windows, однако впоследствии стал своего рода стандартом.

К счастью, сегодня существует сервер разделения файлов CIFS с открытым исходным кодом. Его название – Samba. Кроме того, есть коммерческие продукты, такие как LanManager и NetApp, которые также предоставляют сервисы через этот протокол. В состав FreeBSD входят программы для доступа к разделяемым файлам CIFS. Эти программы существуют в виде двух частей: модуля ядра и нескольких программ userland.

Предварительная подготовка

Перед началом работы с CIFS соберите базовую информацию о сети Windows:

- Рабочая группа или имя домена Windows
- Действительное имя пользователя и пароль
- IP-адрес сервера WINS, или DNS-имена всех хостов, к которым вы хотите получить доступ. (В системе Windows IP-адрес сервера WINS можно получить, запустив `ipconfig /all`.)

Наборы символов

Первая трудность, возникающая при доступе к разделяемым ресурсам Windows, связана с поддержкой множества наборов символов, которой славится Windows. (Пользователи Windows могут очень легко применять символы, которые не присутствуют в стандартном английском алфавите. Надо сделать так, чтобы эти символы не сбивали ядро с толку.)

В состав ядра FreeBSD не входят библиотеки для поддержки множества наборов символов, поэтому такие библиотеки надо добавить. Прежде чем настраивать в ядре поддержку CIFS, установите «порт» */usr/ports/converters/libiconv*.

Поддержка CIFS, предоставляемая ядром

Теперь перекомпилируйте ядро, настроив поддержку CIFS. Для этого добавьте в ядро следующие параметры:

```
options         NETSMB
options         NETSMBCRYPTO
options         LIBMCHAIN
options         LIBICONV
options         SMBFS
```

Функции ядра SMBFS также доступны как модуль, однако поскольку для поддержки SMB вам так или иначе придется пересобирать ядро, эти функции можно скомпилировать статически.

Инструменты SMB

После сборки ядра установите инструменты SMB из */usr/ports/net/smbfs*.[1] Эти инструменты должны быть точно синхронизированы с ядром. Если вы не настраиваете несколько идентичных машин, то в этом случае пакеты будут по большей части бесполезны. Если вы обновляете систему FreeBSD, необходимо обновить «порт». Дополнительные трудности вызывает тот факт, что основной репозитарий исходного кода SMBFS скрывается за очень перегруженным каналом связи в Казахстане. По этой причине я рекомендую хранить distfile где-нибудь в локальной сети, тогда вы сможете легко пересобрать инструменты без скачивания исходного кода из другой части света (впрочем, это зависит от того, где вы находитесь).

Конфигурирование CIFS

Инструменты SMB используют конфигурационный файл – либо *$HOME/.nsmbrc*, либо */usr/local/etc/nsmb.conf*. Все настройки в *nsmb.conf* перекрывают настройки в домашних каталогах пользователей.

Конфигурационный файл разбит на разделы с помощью меток в квадратных скобках. Так, параметры, применимые к каждому соединению SMB, хранятся в разделе `[default]`. Вы можете создать свои разделы, указав серверы, пользователей и разделяемые ресурсы в одном из следующих форматов:

```
[servername]
[servername:username]
[servername:username:sharename]
```

Например, информация, применимая ко всему серверу, расположена в разделе, названном по имени этого сервера. Информация, применимая к конкретному пользователю, хранится в разделе, названном по имени пользователя. А данные, относящиеся только к отдельному разделяемому ресурсу, хранятся под меткой, включающей в себя его имя. Если у вас нет специфичной информации о пользователях или разделяемых ресурсах, то информацию обо всех ресурсах можно разместить в разделе `[servername]`.

Примечание

Nsmb.conf использует значения CIFS. Например, мое имя пользователя в Windows – `mlucas`, но мое имя пользователя в UNIX – `mwlucas`, поэтому в *nsmb.conf* я указываю mlucas.

Ключевые слова nsmb.conf

При назначении конфигурации для разделов можно использовать ключевые слова. Некоторые из них применимы лишь в отдельных разделах. Например, серверы имеют IP-адреса, а пользователи – нет; следовательно, в пользовательский раздел не добавляются ключевые слова, относящиеся к IP-адресам.

[1] Во FreeBSD 4.4 и последующих версиях эти инструменты включены в базовую операционную систему.

При использовании ключевого слова задайте его значение, указав его после знака равенства: *ключевое слово = значение*. Вот ключевые слова.

```
workgroup=string
```

Ключевое слово `workgroup` задает имя домена NT или рабочей группы Windows, к которым требуется получить доступ.

```
addr=a.b.c.d
```

Ключевое слово `addr` задает IP- (или IPX-) адрес сервера SMB с этим именем хоста Windows. Данное ключевое слово может присутствовать только под простой меткой `[servername]`.

```
nbns=a.b.c.d
```

Ключевое слово `nbns` задает IP-адрес сервера имен NetBIOS (WINS). Эту строку можно разместить в разделе `[default]` или в конкретном `[servername]`.

```
nbscope=string
```

Ключевое слово `nbscope` задает область NetBIOS. Если вы не знаете, что такое область NetBIOS, то это значение устанавливать не надо.

```
retry_count
```

Ключевое слово `retry_count` устанавливает, сколько раз клиент SMB будет пытаться взаимодействовать с сервером, прежде чем заключит, что соединение разорвано. Вероятно, значение по умолчанию приемлемо.

```
timeout
```

Параметр `timeout` – это период времени, в течение которого система будет ждать ответа на запрос SMB, прежде чем предпримет повторную попытку. Вероятно, значение по умолчанию здесь также приемлемо.

```
password=string
```

Ключевое слово `password` задает открытый пароль для пользователя или разделяемого ресурса. Если вы должны хранить пароли в *nsmb.conf*, убедитесь, что права на чтение этого файла есть только у root. Хранить пароли в *$HOME/.nsmbrc* – это неверный подход в многопользовательских системах.

Можно зашифровать пароль, запустив `smbutil -crypt`. Перед зашифрованным паролем стоят два знака доллара ($$). Шифрование поможет предотвратить случайный подсмотр пароля, однако злоумышленник легко его расшифрует.

Минимальная конфигурация: поиск адреса по имени

Итак, создадим базовый файл *nsmb.conf*. Как минимум, нам нужна возможность находить хосты, которые входят в интересующую нас рабочую группу и взаимодействуют с сервером имен NetBIOS. Кроме того, на серверах Windows есть пользовательские настройки для разделения файлов, поэтому имя данного пользователя следует разместить в разделе `[default]`:

```
[default]
workgroup=EXAMPLE
nbns=192.168.2.80
username=unix
```

После задания такой информации должны выполняться базовые запросы имен SMB. Для тестирования примените smbutil(1):

```
# smbutil lookup fileserv4
Got response from 192.168.2.80
IP address of fileserv4: 192.168.1.202
#
```

Если запрос выполнен, значит, базовая функциональность SMB настроена.

Другие функции smbutil

Прежде чем монтировать файловую систему хоста Windows, на этом хосте надо зарегистрироваться. (Эти функции smbutil может задействовать только root.)

```
# smbutil login //unix@fileserv4
Password:
Connected to UNIX
#
```

Итак, наш пароль верен. С помощью команды view утилиты smbutil посмотрим, какие ресурсы предоставляет данный сервер:

```
# smbutil view //unix@fileserv4
Password:
Share        Type       Comment
------------------------------
jsmith$    disk
gdonner$   disk
mlucas$    disk
...
```

Получен список всех разделяемых ресурсов сервера SMB.

Теперь, предполагая, что работа завершена, выйдем из системы на сервере:

```
# smbutil logout //mlucas@fileserv4
Password:
Connection unmarked as permanent and will be closed when possible
#
```

Монтирование разделяемого ресурса

Теперь, когда вы завершили исследования, смонтируем файловый ресурс с помощью mount_smbfs(8). Синтаксис будет таким:

```
mount_smbfs //username@servername/share /mount/point
```

На файловом сервере mlucas у меня есть разделяемый ресурс. Чтобы смонтировать его на моей машине FreeBSD как */home/mwlucas/smbmount*, я выполняю такую команду:

```
# mount_smbfs //mlucas@fileserv4/mlucas /home/mwlucas/smbmount
```

Проверим свою работу с помощью df(1).

```
# df
Filesystem                  1K-blocks     Used    Avail Capacity  Mounted on
/dev/ad0s1a                     99183    49105    42144    54%    /
/dev/ad0s1f                   5186362  3091500  1679954    65%    /usr
/dev/ad0s1e                    198399    22816   159712    12%    /var
procfs                              4        4        0   100%    /proc
//MLUCAS@FILESERV4/MLUCAS      128000    54320    73680    42%  /usr/home/mwlucas/
smbmount
#
```

Теперь я могу выполнять базовые операции с документами на разделяемом накопителе, например в Emacs и StarOffice. Жить стало немного лучше.

Другие ключи mount_smbfs

В mount_smbfs есть несколько ключей для управления поведением системы. Ключ `-f` применяется для выбора иного режима доступа к файлам, а ключ `-d` – для выбора другого режима доступа к каталогу. Например, для того чтобы доступ к содержимому монтированного каталога получил только его владелец, надо выполнить команду `mount_smbfs -d 700`. (Таким образом, права доступа UNIX намного строже, чем в Windows, но сейчас речь не об этом.) С помощью ключа `-u` я даже могу изменить владельца, а с помощью ключа `-g` – группу.

Ключ `-I` предписывает mount_smbfs пропускать процесс разрешения имен NetBIOS и применять только имя хоста или IP-адрес, заданный в командной строке.

Ключ `-N` предписывает mount_smbfs считывать пароль из файла настройки, а не выдавать приглашение на его ввод. Это означает, что открытый пароль должен присутствовать в *nsmb.conf* (как говорилось выше).

Ключ `-W` задает новую рабочую группу. Это значение перекрывает настройки *nsmb.conf*.

Файловая система Windows и, следовательно, SMB используют имена файлов без различения прописных и строчных букв, однако в UNIX имена файлов чувствительны к регистру. По умолчанию SMBFS оставляет регистр таким, какой он есть, однако вам это, возможно, не подойдет. С помощью ключа `-c` можно указать mount_smbfs, что необходимо изменить регистр файлов в файловых системах SMB: `-c l` переведет все имена в нижний регистр, а `-c u` – в верхний.

Работая с mount_smbfs, я убедился, что эта программа достаточно гибка, чтобы справиться почти с любой ситуацией в сети Windows. Таким образом,

FreeBSD можно полностью интегрировать с остальными операционными системами в офисе.

Образцы записей nsmb.conf

Внеся изменения в файл *nsmb.conf*, можно посредством различных имен пользователей организовать доступ к разным разделяемым ресурсам. Можно также пропускать разрешение имен NetBIOS для тех или иных хостов. Вот несколько более сложных примеров, в которых применяются конфигурационные записи, определенные ранее.

Предполагается, что все приведенные здесь примеры являются частью файла настройки, в котором как минимум присутствует такая запись:

```
[default]
workgroup=EXAMPLE
nbns=192.168.2.80
username=mlucas
```

Уникальный пароль на автономной системе

Для машины «desktop» с ресурсом, защищенным паролем, подойдет следующий пример. Функция защиты по паролю есть во многих системах Windows 9x.

```
#На моем настольном компьютере есть разделяемый ресурс с отдельным паролем
[desktop:mlucas]
password=$$1725a5038393e12ee
```

Доступ ко второму домену

В данном примере мы пытаемся получить доступ к отдельному домену «development». Для этого домена назначены отдельные имя пользователя и пароль, отличающиеся от заданных по умолчанию:

```
#development - это другой домен NT с разделяемым именем пользователя
[development]
workgroup=EXAMPLE2
username=support
```

Владение файлами в CIFS

Владение файлами при взаимодействии систем UNIX и Windows может быть проблематичным. Например, имена пользователей UNIX, вероятно, не соответствуют именам пользователей Windows. Кроме того, в UNIX применяется не такая схема прав доступа, как в Windows.

Поскольку для доступа к разделяемым ресурсам служит одно имя пользователя Windows, у вас есть доступ ко всем ресурсам Windows, который предоставляет данная учетная запись. Однако на эти ресурсы необходимо назначить соответствующие права доступа UNIX. По умолчанию mount_smbfs назначает новому разделяемому ресурсу те же права, которые применяются для точки монтирования. В нашем примере каталогом */home/mwlucas/*

smbmount владеет mwlucas, а режим доступа – 755. Согласно этим правам доступа, я могу редактировать содержимое данного каталога, а у других пользователей такой возможности нет.

Доступ клиентов Windows к разделяемым файлам UNIX

Поскольку система FreeBSD может получить доступ к разделяемым ресурсам CIFS, с таким же успехом она может предоставлять ресурсы клиентам CIFS (таким как Windows) с помощью Samba. Samba можно найти в */usr/ports/net/samba*. Веб-сайт Samba находится по адресу *http://www.samba.org*, а полезное руководство – по адресу *http://www.linux.org/docs/ldp/howto/HOW-TO-INDEX/howtos.html*. Пакету Samba посвящены целые книги. Мне не удастся осветить эту тему в нескольких абзацах, поэтому здесь обсуждение Samba заканчивается.

Доступ к серверам печати

Печать в UNIX – тема загадочная. Главным образом это объясняется наличием обширного множества аппаратных средств. Первоначально система печати UNIX разрабатывалась для работы с телетайпом, что накладывало специфические требования. Затем эта система была расширена для обслуживания матричных, буквенно-цифровых, струйных, лазерных и всех других принтеров, однако это наследие лежит на ней тяжким бременем. Вот если бы начать разработку системы печати UNIX с нуля и включить в нее только поддержку лазерных и струйных принтеров, то все стало бы намного проще. К сожалению, это нам не по силам.

Lpd

Тем, кому посчастливилось работать с сервером печати на базе UNIX, настройка принтера должна показаться очень простой. Но если вы применяете сервер печати Microsoft, все становится намного сложнее. Чтобы облегчить свою задачу, попросите вашего администратора Windows установить Print Services for UNIX (NT, 2000 и XP).[1]

Стандартный протокол сервера печати UNIX, lpd (line printer daemon, демон строчного[2] принтера) работает на TCP-порту 515. (Кроме того, существует

---

[1] К сожалениею установка Print Services for UNIX подразумевает несколько часов работы администратора Windows. Он должен установить устройство, переустановить последний пакет заплаток от Microsoft (service pack), а затем заново наложить все заплатки, которые должны быть установлены поверх него. Не гневайтесь на администратора, если он откажется. Вы поймете проблемы администратора намного быстрее, чем он окажет вам помощь в ваших затруднениях.

[2] Это неуклюжее название пришло из времен работы с телетайпами. На самом деле lpd умеет работать с каким угодно принтером. – *Примеч. науч. ред.*

программа демона строчного принтера, lpd(8).[1]) Чтобы проверить, предоставляет ли система сервис lpd, выполните telnet к порту 515 сервера печати – аналогично тестированию почтовых и веб-серверов в предыдущих главах. Если вы можете подключиться к этому порту, значит, в системе запущен демон линейного принтера. (Не ожидайте что-нибудь увидеть при установлении соединения – по этому протоколу очень трудно общаться.)

/etc/printcap

Итак, вы знаете, что сервер печати готов к общению по протоколу lpd, и теперь пора настроить систему для взаимодействия с этим сервером. Ключевым элементом является файл */etc/printcap*. Базовая информация о нем представлена в главе 9, но мы рассмотрим этот файл немного более подробно. Вот образец файла *printcap*:

```
❶ lp|SalesPS|ThirdFloorPrinter:\
        ❷ :rp=SalesPS:\
        ❸ :rm=printserver:\
        ❹ :sd=/var/spool/output/lpd:\
        ❺ :lf=/var/log/lpd-errs:
```

Первая строка файла содержит имя принтера, `SalesPS|ThirdFloorPrinter` (❶). Традиционно принтеры UNIX называются «lp», «lp1», «lp2» и т. д.; lp – это принтер по умолчанию; таким образом, принтер `SalesPS` – это наш принтер по умолчанию. В данном примере символ конвейеризации (|) отделяет друг от друга разные имена одного и того же принтера.

Запись `rp=SalesPS` (❷) – это имя принтера, по которому к нему будет обращаться хост, указанный в параметре `rm`. Список имен принтеров надо получить у сетевого администратора. Если это принтер печати Microsoft, можно осведомиться о доступных принтерах, совместный доступ к которым обеспечивает сервер печати. Выберите принтер, который поддерживает PostScript – общий протокол печати. PostScript используют все современные принтеры. Кроме того, некоторые из них предоставляют свои системы печати. К счастью, большинство программ FreeBSD посылают на принтеры либо простой текст, либо текст PostScript.

Примечание

Серверы Microsoft нередко разделяют один принтер под разными именами. Имена применяются для выбора варианта печати. Если в вашем случае так оно и есть, выберите имя, которое соответствует PostScript!

Если вы не знаете имя удаленного принтера, попробуйте lp1, AUTO или PORT1. (Регистр важен!) В то время как lp1 и PORT1 обычно соответствуют PostScript, AUTO пытается определить вид отправляемых заданий на печать.

[1] В оригинале так и написано. Естественно, демон lpd, который реализует соответствующий протокол, сам слушает порт 515. – *Примеч. науч. ред.*

Запись `rm` (❸) – это имя сервера. У вас должна быть возможность выполнить ping до сервера, заданного этим именем.

Запись `sd` (❹) сообщает, в каком месте демон локального принтера будет хранить вывод, который он пытается напечатать.

Наконец, запись `lf` (❺) – это файл протокола принтера. При неполадках с печатью проверьте права доступа на спул-каталог. Владеть им должны root и группа daemon, а код доступа должен быть равен 755 (см. главу 7).

Если вы подставляете правильные значения для имен сервера печати и вашего принтера, то все должно работать верно. Вы можете подставить различные имена, но это необязательно. (Однако в этом случае вы можете запутаться, когда где-нибудь увидите ссылку на «ThirdFloorPrinter»!)

Запуск локального lpd

Подготовив запись в файле *printcap*, вы должны суметь запустить демон lpd(8). Этот демон локального принтера ловит запросы на печать, создаваемые на вашей локальной системе, и отправляет их удаленному демону печати. Демон локального принтера можно запускать во время загрузки, если включить эту возможность в *rc.conf*.

```
lpd_enable="YES"
```

В качестве альтернативы можно просто запустить lpd от имени root.

Тестирование принтера

Протестировать настройку принтера можно с помощью команды lpq(1), которая отразит состояние принтера. Если все работает, вы должны увидеть примерно следующее:

```
# lpq
no entries
#
```

Если в данный момент принтер выполняет печать, вывод будет примерно таким:

```
# lpq
lp is ready and printing
Rank   Owner      Job  Files                                 Total Size
1st    mwlucas    4    (standard input)                      93151 bytes
#
```

Для тонкой настройки принтера обратитесь к информации о */etc/printcap*, представленной в главе 9.

Локальные принтеры

Если сервер печати не предоставляет сервис lpd или же этот сервер напрямую подсоединен к параллельному порту, то с ним другая история. Такой

сервер выглядит дружелюбно и управляется сценарием. Для его работы необходимо установить дополнительное программное обеспечение. Обратите внимание на */usr/ports/print/apsfilter*.

Хорошая новость состоит в том, что apsfilter поможет говорить на языке CIFS с удаленным сервером печати. Этот пакет поможет сконфигурировать локальный принтер, будь то модный лазерный принтер или одна из моделей струйных USB-принтеров El Cheapo стоимостью $59. Если тип принтера неизвестен, то, возможно, придется возвращаться к конфигурационному сценарию несколько раз. Однако после должной настройки все будет работать замечательно.

X: графический интерфейс

Графический пользовательский интерфейс (GUI) в системах UNIX называется X Window System, или X (для краткости). По сравнению с другими оконными системами X имеет преимущества и недостатки. Однако стоит только к ней приноровиться, как X станет весьма полезной.

Одна из любимых мною особенностей X заключается в том, что программа может отображать вывод на другой машине, а не только на той, где она запущена. Я часто запускаю программу на сервере и работаю с ней с помощью пользовательского интерфейса на своей настольной системе. Например, на моем лэптопе установлен только StarOffice. Когда я работаю на одном из серверов в моем подвале[1] и мне надо напечатать жалобу поставщику программ на его непотребный продукт, я подключаюсь к лэптопу в библиотеке наверху и запускаю StarOffice. Хотя эта программа запускается на лэптопе, окно программы отражается на моем локальном терминале. Это может быть даже удобнее, чем сидеть за лэптопом, поскольку принтер, конверты и печати находятся недалеко от серверной комнаты.

Предварительная подготовка X

Прежде чем настроить X, необходимо узнать базовые параметры системы. Откройте руководство по монитору и найдите значения horizontal sync (частота горизонтальной развертки) и vertical refresh [rate] (частота вертикальной развертки). Также изучите руководство по вашей видеокарте и найдите точное название модели и номер.

Версии X

FreeBSD поддерживает версию X под названием XFree86, которая основана на X версии 11 выпуск 6, или X11R6. На момент написания этих строк при-

[1] Прежде чем удивиться наличию в моем доме серверной комнаты, знайте, что моя жена называет подвальную серверную комнату «свалкой», а машины, находящиеся там, – «Франкенштейнами». Я не вижу ничего плохого в использовании 166 MHz Digital Alpha для подогревания чайника, но моя жена придерживается другого мнения по этому поводу.

менялись две различные версии XFree86. Версия 3.3 – это более старший стабильный стандарт, а версия 4 – новая и многообещающая. Версия, которая будет у вас, зависит от настроек по умолчанию, заданных в установленной системе FreeBSD.

Если у вас есть одна версия, но вы хотите другую, то можете деинсталлировать все «порты», которые применяют X, удалить все дерево каталогов */usr/X11R6* и установить «порт» необходимой версии */usr/ports/x11*. (Такой процесс потребует продолжительного времени и займет большое дисковое пространство.) Затем надо переустановить все «порты», зависящие от X. Пользуйтесь.

Конфигурирование X

Для конфигурирования X можно использовать текстовый инструмент xf86cfg либо графические инструменты xf86config (версия 4) или xf86Setup (версия 3). Будьте внимательны: в отличие от всех других операций на вашем компьютере, неправильное использование XFree86 может причинить физический ущерб монитору и видеокарте. Если при тестировании конфигурации XFree86 ваш экран вспыхивает, стробирует или проявляются другие неполадки, немедленно покиньте X. (Закрыть X можно с помощью комбинации клавиш Ctrl-Alt-Backspace.)

Делаем X приличной

По умолчанию X ужасна. Действительно ужасна. Она ужасна потому, что обслуживает только базовую отрисовку экрана. Это не имеет значения, пока отображаются картинки не сложнее терминальных окон, меню и т. д., но X не сможет нарисовать очень хорошие картинки. Чтобы X была действительно удобна, необходима настольная среда или оконный менеджер.

Тема оконных менеджеров – это еще один предмет спора в среде пользователей UNIX. Встречаются люди, считающие лучшим оконным менеджером в мире пакет, поставляемый вместе с X, – twm(1). Должен признать, что он не так уж плох на допотопных машинах. Если бы у меня была система с оперативной памятью размером 4 Мбайт, то я бы выбрал twm.

Но в моем компьютере оперативной памяти больше, чем 4 Мбайт, и я предпочитаю оконные менеджеры с дополнительными функциями. В */usr/ports/x11-wm* можно найти довольно много оконных менеджеров. Поэкспериментируйте с ними. Я остановлюсь на четырех менеджерах. Мой выбор обусловлен их популярностью и моим опытом работы с ними.

Популярные настольные среды X

Лично я предпочитаю оконный менеджер WindowMaker (*/usr/ports/x11-wm/windowmaker*). Он быстрый, легкий, отлично настраиваемый, поддерживает наборы настроек и отлично выглядит на мониторе чуть ли не любого размера.

Оконный менеджер fvwm2 (*/usr/ports/x11-wm/fvwm2*) – это классика, которая по-прежнему широко используется. Она очень гибка, а вся конфигура-

ция представлена текстовым файлом. Это стабильная, надежная программа, занимающая мало памяти. И еще она скучна.

В среде новых пользователей UNIX нибольшей популярностью пользуются настольные среды Gnome (*/usr/ports/x11/gnome*) и KDE (*/usr/ports/x11/kde2*). Обе они пытаются угодить всем пользователям. В обеих есть инструменты системного конфигурирования, веб-броузеры и офисные пакеты. По моему мению, обе среды излишне надуты и глупы – подобно лощеным коммерческим системам, которые они надеются заменить. Однако если вы предпочитаете интерфейс в стиле Windows или Mac, вам может подойти любая из этих сред.[1]

Выбор оконной среды – дело глубоко личное. Я думаю, что было бы интересно провести психологическое исследование пользователей UNIX, классифицируя их по выбору оконного менеджера. (Полезно? Нет. Зато интересно.) Поэкспериментируйте с несколькими из них, и вы быстро найдете оконный менеджер, соответствущий вашему стилю работы.

Настольные приложения

Если оконный менеджер выбран, то можно приступать к работе с настольными приложениями. Стандартным настольным приложением UNIX попрежнему является командная строка. Для окон X существует специальный тип терминального окна, называемый xterm. В состав некоторых оконных менеджеров входят свои терминальные приложения. Кроме того, для X были написаны и другие терминальные программы, но xterm есть в любой среде.

Однако вы установили X не только для того, чтобы работать в терминальном окне. Рассмотрим базовые приложения, которые могут вам понадобиться.

Веб-броузеры

В ближайшее время вы не найдете Internet Explorer для FreeBSD. Вероятно, это хорошо. В состав FreeBSD входят несколько других веб-броузеров. Из них наиболее известен Netscape.

Netscape

В */usr/ports/www* можно найти несколько разных версий Netscape. В состав «коммуникатора» (communicator) входят почтовый клиент, программа чтения новостей и инструмент для создания веб-страниц; версии «навигатора» (navigator) включают в себя только веб-броузеры.

В системе безопасности версий Netscape для FreeBSD и BSDi есть прорехи, на которые Netscape не удосуживается обратить внимание. Если вам нужен Netscape, применяйте версии для Linux. В них реализована поддержка до-

[1] Фанаты Gnome и KDE могут сказать, что в моей предпочтительной среде отсутствуют важные функции. Все определяется тем, что вам нужно от настольной среды.

полнительных модулей Linux (plug-ins), более распространенных, чем дополнительные модули FreeBSD, но требующих установки модуля совместимости с Linux (см. главу 11). Кроме того, Linux-версия броузера позволяет задействовать дополнительные модули Linux RealPlayer и Shockwave.

Mozilla

Если вам нужен броузер поновее Netscape, обратите внимание на Mozilla (*/usr/ports/www/mozilla*). На этом коде основан Netscape 6. Броузер Mozilla солиден, надежен и улучшается с каждым днем. Он также поддерживает Java и Flash. Я предпочитаю именно его. «Порт» броузера Opera для Linux (*/usr/ports/www/linux-opera*) набирает популярность, и многие, сравнивая Netscape и Opera, говорят о последнем благосклонно. Еще один распространенный выбор – konqueror. Это менеджер файлов, входящий в состав оконного менеджера KDE. Он умеет просматривать как локальные файлы, так и веб-страницы. Он устанавливается вместе с KDE.

Текстовые броузеры

Если вы не установили X, то по Сети можно бродить при помощи текстового броузера. Именно так бродили по Сети в начале 1990-х. Нередко я прибегаю к текстовому броузеру, если мне надо быстро получить информацию без графики и других прибамбасов. Вот три наиболее популярных текстовых броузера: Lynx (*/usr/ports/www/lynx-ssl*), w3m (*/usr/ports/www/w3m*) и links (*/usr/ports/www/links*). Применение одного из этих броузеров позволит приобрести поучительный опыт; например, много слепых людей используют эти инструменты для доступа в Интернет.

Почтовые программы

FreeBSD поддерживает много различных почтовых программ (см. */usr/ports/mail*). А разные люди предпочитают работать с разными почтовыми клиентами, поэтому надо выбрать какой-нибудь из нескольких. Перечислим наиболее популярные почтовые программы.

Почтовые программы с GUI

Тем, кто пришел из мира Windows или Mac, вероятно, покажутся удобными Arrow (*/usr/ports/mail/arrow*) или Evolution (*/usr/ports/mail/evolution*). Программа Evolution похожа на Microsoft Outlook, но требует значительных ресурсов настольной среды Gnome. Кроме того, можно использовать почтовый клиент, входящий в состав Netscape (*/usr/ports/www/linux-netscape47-communicator*) или Mozilla (*/usr/ports/www/mozilla*). Пользователям, пришедшим из среды GUI, эти программы покажутся простыми и интуитивно понятными.

Mutt

Если вам нужна почтовая программа, которая больше соответствует стилю UNIX, попробуйте Mutt (*/usr/ports/mail/mutt*). Это результат 15-летней

эволюции почтовых программ. Все опытные пользователи UNIX, которых я знаю, остались довольны Mutt.[1] Эта программа хорошо конфигурируется и позволяет пользователю очень быстро обрабатывать информацию.

Одна из замечательных особенностей Mutt состоит в том, что сообщения можно группировать по темам (нитям); таким образом, сообщения, являющиеся частью отдельной дискуссии, группируются вместе, по порядку. Я получаю гигантское количество писем, порой свыше 2000 сообщений в день. Каждое письмо надо прочесть и оценить. Лишь часть из них требует моего внимания и только на пару десятков писем надо написать ответ. (Обычно таких писем немного больше, но я никогда не могу найти на них время.) Если бы я пользовался почтовой программой вроде Microsoft Outlook, дела мои были бы плохи. Mutt обеспечивает мне возможность почти разумного управления этим валом.

Pine

Другая популярная почтовая программа UNIX – программа Pine. Она выглядит более дружелюбно, чем Mutt. В ней есть меню. Хотя программа Pine не настолько конфигурируема, как Mutt, она является хорошим выбором для людей, далеких от техники, а также для тех, кто не обрабатывает большие объемы электронной почты. Моя жена с удовольствием читает письма в Pine.

Офисные пакеты

Если вы работаете на настольном компьютере, вам понадобятся текстовый редактор, электронные таблицы, программа рисования и т. д. Такие программы есть во FreeBSD, однако для работы с ними необходимо установить X.

StarOffice

StarOffice (*/usr/ports/editors/staroffice60*) – это полный офисный пакет с множеством функций. (На самом деле это версия для Linux, работающая в режиме Linux.) StarOffice довольно эффективен; в нем была написана эта книга, и данная версия StarOffice стоит ничтожно мало.

OpenOffice

Компания, создавшая StarOffice, была куплена компанией Sun Microsystems, решившей опубликовать исходный код программы. Это означало, что они должны были убрать оттуда фрагменты кода, лицензированные другими компаниями, и заменить их. В результате появился пакет OpenOffice (*/usr/ports/editors/openoffice*). Предполагается, что OpenOffice запускается на FreeBSD как в «родной» системе. Однако пройдет какое-то время, прежде чем все функции StarOffice станут доступны. Среднему пользователю я настоятельно рекомендую OpenOffice версии 1.0 или более поздней.

---

[1] Сейчас мой почтовый ящик заполнится сообщениями от людей, которые скажут, что их седая борода длиннее моей и что программа Mutt им не понравилась. Вполне возможно.

koffice

Другой популярный выбор – koffice (*/usr/ports/editors/koffice*). Однако он собирается вместе с KDE, поэтому кто-то может счесть его системные требования слишком высокими.

Музыка

Каждый пользователь, проработавший на компьютере хоть какое-то время, проигрывает компакт-диски. Акустическая система компьютера не соответствует стандартам меломанов, но работает достаточно хорошо.

В состав FreeBSD входят различные проигрыватели CD. Все они находятся в */usr/ports/audio*. В Gnome и KDE есть свои программы. Существует несколько утилит, разработанных для WindowMaker. Из них я отдаю предпочтение */usr/ports/audio/wmcdplay*. Многим нравятся xmms и gkrellm.

В указанном каталоге можно найти множество проигрывателей CD, MP3, MIDI и MPEG.

Графика

В современном компьютерном мире применяются различные графические форматы, а попытка просмотреть файл незнакомого формата может стать причиной сильного огорчения. Здесь будут представлены некоторые из наиболее распространенных графических форматов, а также программы, необходимые для их просмотра.

Просмотр типичной графики

Большинство файлов графических форматов можно просмотреть с помощью xv (*/usr/ports/graphics/xv*). Программа xv понимает следующие форматы файлов (список не полон): GIF, JPEG, TIFF, PBM, PGM, PPM, X11 bitmap, BMP, XPM, PCX, IRIS RGB и PNG.

Другой типичный формат – PostScript или Encapsulated PostScript. Эти файлы можно просмотреть с помощью ghostview (*/usr/ports/print/ghostview*).

Столкнувшись с незнакомым графическим форматом, обратитесь к */usr/ports/graphics/README.html*. Есть шансы, что здесь будет указана маленькая программа, способная открыть этот файл.

Просмотр видео

Чтобы просмотреть видео, попробуйте xanim (*/usr/ports/graphics/xanim*) или xmps. «Порт» avifile может проигрывать видео MPEG4/DIVX.

Создание графики

Для создания собственной графики предназначен пакет Gimp (*/usr/ports/graphic/gimp*). Этот набор инструментов сравним с Adobe Photoshop.

Настольные утилиты

В наши дни люди размещают на рабочем столе компьютера все, что угодно, начиная от листков для заметок и заканчивая калькуляторами. Лично я предпочитаю настоящие клеящиеся листки из бумаги – их можно снять с монитора и передать кому-нибудь другому. Однако если вам нужна такая функциональность, FreeBSD ее предоставит. Полный список утилит можно составить по каталогу */usr/ports/deskutils*. Рассмотрим здесь наиболее яркие.

GnuCash

Одна из бесплатных прикладных программ-«приманок»[1] – GnuCash, персональный менеджер денежных операций, подобный Quicken. Его можно найти в */usr/ports/deskutils/gnucash*. Для его работы необходима среда Gnome, но его можно использовать совместно с любым другим оконным менеджером. GnuCash еще не может взаимодействовать с банковскими финансовыми системами, как это умеет Quicken, однако на момент выхода этой книги такая возможность должна появиться.

Palm

Аналогично, персональные цифровые помощники довольно типичны в наши дни. Устройства WinCE слишком привязаны к операционной среде Microsoft, чтобы хорошо работать с FreeBSD. Однако с FreeBSD отлично работают устройства Palm, включая Handsprings. Все программное обеспечение Palm можно найти в */usr/ports/palm*.

Игры

А теперь важная информация! В состав FreeBSD входит множество различных игр. Все они расположены в */usr/ports/games*. Если вы запускаете KDE или Gnome, то несколько простых игр у вас уже установлено. Эти пакеты без сопутствующих оконных менеджеров можно найти как в */usr/ports/games/kdegames2*, так и в */usr/ports/games/gnomegames*. «Порты» автоматически установят минимальные базовые библиотеки, необходимые для запуска этих игр. Многим из нас придется поступить именно так, поскольку эти библиотеки нужны другим программам.

Xevil

Если вы хотите наносить увечья, играть по сети и действовать быстро, обратите внимание на xevil. Это старая добрая двухмерная «стрелялка» с лестницами, роботами и всей необходимой экипировкой для того, чтобы быть застреленным, сожженным, взорванным. Эта простая игрушка не для обидчивых, но выбрав ее, можно провести за ней несколько дней. Лучше всего то, что она легковесная и для ее запуска не нужны ни Gnome или KDE, ни qt или xview, ни любые другие графические инструменты.

---

[1] Предназначены для привлечения потенциальных клиентов. – *Примеч. перев.*

Heretic, Doom и Quake

Если вы любитель трехмерной графики, обратите внимание на «родные порты» FreeBSD игр Heretic, Doom и Quake (*/usr/ports/games/quakeforge*). Это классические игры, выпускаемые Id Software: беги, добудь более мощное оружие, найди злых монстров и накорми их ракетами, которых они заслуживают. Для Doom характерна одна трудность – эта программа запускается только на X-терминалах с восьмибитным цветом (256 цветов).

Можно запустить сессию X, которая подойдет для Doom. Для этого надо выполнить команду:

```
# startx -bpp 8
```

Это означает, что вы должны выйти из X и запустить ее снова, однако это не так уж плохо. В конце концов, попробуйте запустить Doom на современной системе Windows и посмотрите, понравится ли вам увиденное.

Nethack и Angband

Тот, кто считает, что xevil и Doom слишком насыщены графикой, может вернуться в 1980-е вместе с Nethack, Angband и их вариантами. Это простые игрушки с прямоугольной плоской графикой. Вы – герой в темнице, пытающийся достичь самого низкого уровня и расправиться с ужасными тварями. Персонажи в каждой версии игры разные.

Nethack запускается без X, однако к этому привыкаешь. Angband запускается в X и, по сравнению с Nethack, содержит больше графики. Обе игрушки постоянно разветвлялись, поэтому тут есть из чего выбрать. Лично я предпочитаю стандартную игрушку Angband, но на Nethack я тоже потратил несколько часов.

Цивилизация

Если насилие вам не по вкусу, отправляйтесь прямо к Freeciv (*/usr/ports/games/freeciv*). Freeciv – это клон игры «Цивилизация» (Civilization). Вы руководите поселением колонистов и пытаетесь создать мировую империю. Можно соревноваться по сети с другими игроками или играть в одиночку. Из всех ловушек, подстерегавших меня на пути к сдаче книги в срок, Freeciv, безусловно, была самой коварной и опасной.

Если вы предпочитаете оригинальную программу, а не клон, запустите демо-версию Civilization 2. Обратитесь к */usr/ports/games/civ2demo*.

Minesweeper, Pac Man, Mahjongg и др.

Вообще говоря, приятно иметь под рукой и другие игрушки. Когда я начал заниматься технической поддержкой, я играл в Minesweeper, чтобы убить время, пока пользователь бубнит о проблеме недели. Есть несколько клонов Minesweeper, например freesweep, wmtimebomb и yamsweeper. Клон Pac Man можно найти в *xchomp*. Игра Mahjongg представлена как xmahjongg или xvmahjongg. Есть шахматы, го и чуть ли не все классические игры, которые занимали умы человечества последнюю тысячу лет.

Послесловие

Если вы дошли до этого места, то уже знаете, как управлять системой FreeBSD и применять ее в качестве платформы для выполнения любых серверных задач. Возможно, вам потребуется изучить работу новых программ, но сейчас вы знаете об операционной системе достаточно, чтобы настроить ее по своему усмотрению. Поздравляю! FreeBSD – это замечательная, гибкая платформа. Она способна играть любую роль в вашей сети. В заключение я кратко раскажу о некоторых других аспектах FreeBSD.

Во FreeBSD две составляющих. В этой книге внимание было сосредоточено на программном обеспечении, которое образует операционную систему. Другая часть – это сообщество, создающее эти программы.

Сообщество

Сообщество FreeBSD состоит из специалистов по вычислительной технике, программистов, пользователей, системных администраторов, технических писателей и всех интересующихся этой системой. У них разное общественное положение, уровень образования. Они разбросаны по всему миру. Лично я общался с пользователями и разработчиками FreeBSD из Соединенных Штатов, Канады, Мексики, Великобритании, России, Казахстана, Дании, Польши, Австралии и Японии. Некоторые специалисты, с которыми я работал, являются выходцами из других стран, однако в сообществе FreeBSD национальность не имеет значения. Не сомневаюсь, что это люди различных рас и вероисповеданий, но в онлайне это неважно. Некоторые из них – врачи. Другие – специалисты по вычислительной технике. Третьи работают в пунктах проката видеокассет. Однажды я тесно сотрудничал с блестящим разработчиком, который оказался подростком. Поскольку большая часть взаимодействия в сообществе происходит в онлайне, вас представляют только ваши слова и ваша работа.

Порой в сообществе возникают конфликты, особенно по поводу будущего развития программного обеспечения FreeBSD, однако в целом обстановка не настолько беспокойная, как это можно было ожидать от тысяч человек с различным прошлым.

Это люди, которые улучшают FreeBSD, продвигают эту систему и делают ее чем-то большим, нежели набором нулей и единиц.

Каждый человек вносит свой вклад по своим причинам. Крошечная часть разработчиков получают деньги за улучшение кода. Средства поступают либо от организаций, зависящих от FreeBSD, либо от правительственных организаций, таких как DARPA. Для большинства разработчиков FreeBSD эта

деятельность является хобби, поэтому они могут программировать более раскрепощенно, чем это позволено им на основной работе. Сроки выхода версий FreeBSD анонсируются на месяцы и годы вперед, поэтому разработчики могут сопоставить их со своей занятостью и определить свой вклад.

Многие из нас не являются разработчиками, но вносят свой вклад в разные части FreeBSD. С тех пор как я начал писать эту книгу, я стал участником Проекта документирования (the Documentation Project). Эта группа отвечает за написание инструкций. Код может быть прекрасным, но если никто не знает, что с ним делать, система будет бесполезной! Другие специалисты отвечают на вопросы в почтовых рассылках или обслуживают сайты поддержки. Мы делаем это ради удовлетворения и удовольствия, мы хотим отдать то, что когда-то дали нам.

Вы можете просто применять систему FreeBSD по своему усмотрению. Однако со временем у многих из нас возникает желание внести свой вклад в деятельность сообщества. Вот почему сообщество растет, а рост сообщества означает, что развитие FreeBSD будет продолжаться.

Что вы можете сделать?

Для тех, кто по тем или иным мотивам хочет оказывать помощь, работы много. Чтобы внести свой вклад во FreeBSD, необязательно писать код. Время от времени кто-нибудь пишет в почтовой рассылке: «Я рад помочь, но я не умею программировать». Подобные причитания попадаются мне в почтовых рассылках FreeBSD с тех пор, как я начал читать их в 1996 году. Вероятно, они появлялись и ранее. Обычный ответ на такие сообщения – тишина. В конце концов, если вы уже решили, что не в состоянии помочь, то не о чем и говорить.

Никто не отрицает, что программисты – это главные герои FreeBSD. Многие из них обладают впечатляющими навыками, а большинство из нас даже не мечтают стать следующим Брюсом Эвансом (Bruce Evans). Мэтт Диллон (Matt Dillon) по праву заслужил почет и уважение за его великолепный ответ на сообщение об ошибке от Apple Computer. Каждая команда BSD хранит свои истории о героизме в программировании. Но даже тот, кто не может запрограммировать выход из мокрого бумажного пакета, тем не менее, может оказать помощь. Основной вопрос: «Что вы можете делать?» Не «Что нужно проекту», не «А что, если моя любимая ОС будет такой-то?», не «Какие функции вам нужны?» Какими навыками вы обладаете сейчас? Есть шансы, что ваши навыки пригодятся в любом проекте BSD или в любых других проектах по разработке бесплатного программного обеспечения.

Например, в течение многих лет я оказывал техническую поддержку. Я потратил несколько лет, совершенствуясь в программировании, большей частью для того, чтобы сменить сферу деятельности. Хотя я пошел вверх и сегодня опасно близок к стезе руководителя, я по-прежнему интересуюсь тем, что находится у компьютеров внутри. Мастерство администрирования операционных систем происходит от чтения кода. Много лет я был убежден, что

не смогу внести свой вклад в развитие FreeBSD, пока не научился кодировать и не познакомился с Тайными Мастерами Ядра.

Кроме того, я пишу слишком много, и это получается у меня хорошо. Однажды я решил написать техническую статью о FreeBSD, и один журнал ее у меня чуть с руками не оторвал. С тех пор я написал сотни страниц о FreeBSD. Люди знают мое имя. Меня уважают за мой вклад, и я не написал ни одной строки программного кода.

В чем вы сильны? Задействуйте свои навыки. Это будет оценено по достоинству.

Если ничего больше...

Если у вас на самом деле нет полезных навыков или идей, просто перечитайте эту книгу и FreeBSD FAQ, подпишитесь на *FreeBSD-questions@FreeBSD.org* и помогайте другим пользователям. Так начинал я. Если вы изберете такой путь, вежливо направляйте людей к существующим информационным ресурсам. Когда кто-нибудь задает вопрос, ответ на который есть в FAQ, предоставьте ему ссылку на основную страницу FAQ. Если данный вопрос задавался ранее, посоветуйте обратиться к архивам почтовых рассылок. Если вы научите людей помогать самим себе, мы сможем снизить трафик почтовой рассылки. (Я подписан на 13 почтовых рассылок FreeBSD. Должен сказать, что снижение трафика почтовых рассылок будет полезно.) Как говорит старая пословица, научите пользователей ловить рыбу, и мы продадим рыбные крючки. Отвечая на вопросы в почтовых рассылках, вы увидите, в чем нуждается Проект и какой вклад вы сможете внести.

Увидев задачу, которую вы можете решить, остановитесь на минуту и поразмышляйте. Если необходимо, сделайте заметку и вернитесь к ней на следующий день. Действительно ли это вам по силам? Неважно, насколько проста и мала задача; сможете ли вы вооружиться профессиональными инструментами, сесть и выполнить ее?

Получение результата

Открою большой секрет получения результата во FreeBSD. Все компоненты появились так: кто-нибудь видел ту или иную потребность, которую он мог удовлетворить, и делал это. NetBSD и FreeBSD ведут начало с того времени, когда группа пользователей 386BSD устала ждать очередного выпуска. Майк Смит (Mike Smith) внедряет поддержку ACPI (Advanced Configuration and Power Interface) во FreeBSD 5.0, поскольку считает это важным. Я не запрашивал разрешения у проекта FreeBSD на написание статей об этой системе. Вы не должны просить разрешения отвечать на сообщения в почтовых рассылках и вносить свой вклад в проект FreeBSD. Если ваши задумки достаточно сложны и вам хочется услышать чье-то мнение, прежде чем приступить к работе, обратитесь к почтовой рассылке.

Первым делом поищите в архивах почтовой рассылки сообщения, имеющие отношение к вашей идее. Почитайте предыдущее обсуждение этой темы.

Многие проекты предлагались, но не были реализованы. Есть вероятность, что данная идея приходила в голову кому-то другому и была заброшена. Если вашу идею уже кто-то обсуждал и одобрил, то можно предположить, что вы получили бы такой же ответ. Если идея обсуждалась настолько давно, что проект кажется мертворожденным, приступайте к работе.

Например, время от времени в почтовой рассылке FreeBSD-hackers появляются пожелания, что файловую систему JFS (Journaled File System, журналируемую файловую систему) компании IBM следует перенести во FreeBSD. Как правило, это признается Хорошим Делом. Затем автор сообщения исчезает и больше о нем ничего не слышно. Перенос файловой системы подразумевает большой объем работы и требует особых навыков. Не удивительно, что многие люди брались за эту задачу и не достигали результата. Однако вам не надо посылать еще одно сообщение с вопросом, насколько хороша эта идея. Если у вас есть навыки для решения этой задачи, приступайте! Вы доставите себе удовольствие, если разместите во FreeBSD-hackers такое сообщение: «По такой-то ссылке вы можете найти мои заплатки для переноса JFS во FreeBSD. Можно монтировать накопители, создавать файлы, но при создании каталогов проявляются странные эффекты. Они могут разрушить ваши данные, вывести из строя ядро и причинить другие неприятности. Буду рад любым комментариям.»

Иное мнение

Если кто-нибудь уже разбил вашу идею в пух и прах, а вы хотите услышать другие мнения, задайте вопрос в почтовой рассылке. Если вы действительно хотите привлечь внимание, ваше сообщение должно быть очень кратким и по существу. Многие полезные идеи можно описать в одном предложении или, самое большее, в двух; все остальное – детали реализации. Помните, что некоторые специалисты получают свыше двух тысяч писем в день. Простое сообщение с вопросом: «Эй, кто-нибудь считает это полезным?» – отличная граница между тупой тратой времени и серьезной необходимостью.

Избегайте замечаний, которые сводятся к следующему: «Почему этого никто не делает?» Большинство таких замечаний относится к одной из трех категорий: очевидной («А что будет, если мы встроим процессор в мой автомобиль?»), глупой («Почему в ядре нет параметра ПРОЧТИМОИМЫСЛИИДЕЛАЙТОЧТОЯХОЧУ?») или смешанной («Почему система не поддерживает мой Sinclair ZX80?»). В любом из этих случаев вопрошающий почти всегда совершенно не подготовлен к реализации таких функций и даже не предлагает купить напитки тем, кому такие задачи по силам. Как правило, вы можете быть уверены, что квалифицированные специалисты рассмотрят эти идеи и либо помогут в их реализации, либо будут работать над более важными задачами.

Не следует ожидать, что тысячи специалистов ответят: «Это отличная идея». Почтовые рассылки – это место для общедоступных обсуждений. Если кто-либо согласен с тем или иным сообщением, он не будет писать: «Согласен!» Однако если несколько человек согласятся с вами и никто не скажет,

что «ваша концепция настолько ужасна, что дальше некуда», то вы можете посчитать свою идею приемлемой. Это равносильно приобретению «пакета акций» FreeBSD. Такие отклики можно считать полным одобрением.

Изредка кто-нибудь может ответить вам так: «Я думаю, что над этим работает Фред; не написать ли вам ему?» Возможно, вы присоединитесь к уже существующей разработке. Другие специалисты захотят обсудить с вами детали реализации. Это можно считать одобрением; они не станут обсуждать детали, если сама идея им не нравится.

Те же рекомендации применимы к каждой части проекта. Сможете ли вы перевести документацию на санскрит? Да. Можете ли вы организовать группу пользователей в Топеке?[1] Разумеется. Если такой проект не существует, приступайте к его реализации.

Вперед!

Возможно, это самая важная часть. Войдя в проект, приступайте к работе. О проекте можно говорить целый день, но это бесполезное времяпрепровождение, если в результате ничего не создается. Большинство идей, встречавшихся в рассылках FreeBSD, провалились на этапе планирования. Есть большой объем задач, которые можно решить. За них никто не берется, поэтому все остается так, как есть. Все системы BSD, да и все проекты по разработке программ с открытым кодом страдают от этого в той или иной степени. Непрограммисты могут очень помочь, обнаружив маленькую «дыру» и заделав ее. Некоторые специалисты организуют независимые веб-сайты. В качестве примера можно назвать замечательный сайт FreeBSD Diary (*http://www.FreeBSDDiary.org*) Дэна Лэнгилла (Dan Langille). Одни члены сообщества непосредственно улучшают программы, которые выпускает Проект. Другие известны как «пижоны, которые околачиваются в почтовых рассылках и помогают пользователям настроить ppp». Все это абсолютно важно. Вы помогаете проекту FreeBSD процветать и расти.

Я предвкушаю нашу встречу в почтовых рассылках.

---

[1] Topéka – город в США, штат Канзас. – *Примеч. перев.*

Приложение

Полезные sysctl MIB

Это приложение представляет собой словарь полезных sysctl MIB. Инструменты для управления MIB обсуждаются в главе 4. (Когда MIB подробно рассматривается в том или ином месте книги, приводится ссылка на соответствующую главу.) Безусловно, в вашей системе есть намного больше sysctl, однако с представленными здесь параметрами я сталкиваюсь ежедневно.

Совет

Необдуманное применение sysctl может легко причинить ущерб работающей системе. Так, если установить на использование ресурсов предел, превышенный уже занятыми ресурсами, в системе может произойти впечатляющая авария. Перед настройкой sysctls убедитесь, что вы представляете последствия своих действий.

Во время подготовки этого приложения я испытывал искушение описать только те sysctls, которые смогут безопасно использовать начинающие администраторы. Однако такой подход значительно уменьшил бы пользу от этого раздела. Вместо этого я прошу вас помнить об опасностях sysctl. Если вы не понимаете назначение того или иного sysctl, не экспериментируйте с ним!

Так, некоторые sysctls позволяют регулировать операции с виртуальными файловыми дескрипторами. Попытки тонкой настройки виртуальных дескрипторов наверняка нанесут ущерб производительности, но иногда такая настройка необходима. Точно так же, не настраивайте систему виртуальной памяти, если вы не понимаете ее работу! Прежде чем экспериментировать с тем или иным sysctl, поищите его описание на страницах руководства и в архивах почтовых рассылок.

Примечание

Некоторые sysctls описаны как «переключатели» (toggles). В этом случае значение 1 предписывает sysctl предоставлять описанный сервис. Если значение равно 0, то сервис отключен.

Без дальнейших предисловий рассмотрим sysctls. Для каждого sysctl представлено типичное значение и описание его работы.

```
kern.maxvnodes: 20973
установка: /boot/loader.conf
```

Максимальное количество *виртуальных файловых дескрипторов* (*vnodes*, дескрипторов виртуальной файловой системы), которые могут быть одновременно открыты в системе.

```
kern.maxproc: 1044
установка: /boot/loader.conf
```

Максимальное количество процессов, которые система может запустить одновременно.

```
kern.maxfiles: 2088
установка: runtime
```

Максимальное количество файлов, которые система может открыть одновременно.

```
kern.argmax: 65536
установка: read-only
```

Максимальное количество байтов, которые можно задать в аргументе execve(2). По существу, это максимальное количество символов, которые можно указывать в одной командной строке. При необычных обстоятельствах этот предел можно превысить. В таком случае обратитесь к xargs(1).

```
kern.securelevel: -1
установка: runtime or /etc/rc.conf
```

Текущий уровень безопасности ядра. См. главу 7.

```
kern.maxfilesperproc: 2088
установка: runtime
```

Максимальное количество файлов, которые может открыть один процесс.

```
kern.maxprocperuid: 1043
установка: runtime
```

Максимальное количество процессов, которые может запускать один пользователь.

```
kern.dumpdev: /dev/ad0s1b
установка: /etc/rc.conf
```

Имя устройства свопа, на которое сохраняется дамп ядра при панике, как это задано командой dumpon(8) во время загрузки. Раздел свопа должен быть больше емкости физической памяти или равен ей. Дамп и паника подробно рассмотрены в главе 20.

```
kern.ipc.somaxconn: 128
установка: runtime
```

Максимальное количество новых соединений, которые система может принять одновременно. Значение по умолчанию равно 128. На сильно нагруженном сервере его можно увеличить до 512 или даже до 1024.

```
kern.ipc.maxsockets: 2088
установка: /boot/loader.conf
```

Общее количество сокетов, доступных в системе.

```
kern.logsigexit: 1
установка:  runtime
```

Завершаясь аварийно, программа обычно посылает сигнал. Если установлен этот переключатель, то имя программы и сигнал завершения будут протоколироваться в */var/log/messages*.

```
kern.init_path: /sbin/init:/sbin/oinit:/sbin/init.bak:/stand/sysinstall
установка: /boot/loader.conf
```

Init(8) – это процесс, который запускает систему. Если вы причинили ущерб системе (скажем, обновление кода прошло неудачно), то с помощью этого sysctl можно указать другой путь к программе init. Однако это может означать, что вы находитесь в плохой форме.

```
kern.module_path: /;/boot/;/modules/
установка: runtime
```

Путь к каталогам, в которых kldload(8) ищет модули ядра.

```
kern.timecounter.method: 0
установка: runtime
```

Во FreeBSD есть два метода, позвляющих определить период времени, прошедший с момента запуска системы. Один из них чрезвычайно точный, но требует больше системных ресурсов. Другой быстрее, но менее точен. Различие между результатами двух методов измеряется в миллисекундах, однако в приложениях, требующих повышенной точности, эти миллисекунды играют большую роль. Присвойте этому sysctl значение 1, если нужен медленный, но суперточный метод. Значение по умолчанию подходит почти для всех приложений.

```
kern.coredump: 1
установка: runtime
```

Этот переключатель управляет дампами ядра. Если установлено значение 1, то при панике ядро выполняет дамп памяти. В главе 20 рассазано, что делать с этим дампом.

```
kern.quantum: 100000
установка: runtime
```

Максимальное количество микросекунд, в течение которого может выполняться процесс, если другие процессы ожидают предоставления времени процессора. Вероятно, это значение лучше не менять.

```
kern.filedelay: 30
установка: runtime
```

Определяет, как часто система синхронизирует файловые данные, хранящиеся в буфере виртуальных дескрипторов и на диске. Этот sysctl предназначен только для опытных системных администраторов!

```
kern.dirdelay: 29
установка: runtime
```

Определяет, как часто система синхронизирует данные о каталогах, хранящихся в буфере виртуальных дескрипторов и на диске. Этот sysctl тоже предназначен только для опытных системных администраторов!

```
kern.metadelay: 28
установка: runtime
```

Определяет, как часто система синхронизирует метаданные файловой системы, хранящиеся в буфере виртуальных дескрипторов и на диске. Этот sysctl тоже предназначен только для системных администраторов!

```
vm.v_free_min: 582
установка: runtime
```

Максимальное количество страниц кэша и свободной памяти, которые должны быть доступны, прежде чем процесс, ожидающий памяти, будет разбужен.

```
vm.v_free_target: 2513
установка: runtime
```

Общее количество страниц свободной памяти и кэша, которое пытается хранить или превысить менеджер виртуальной памяти.

```
vm.v_free_reserved: 185
установка: runtime
```

Если количество страниц свободной памяти падает ниже этого резервного значения, то запуск процесса предпишет менеджеру виртуальной памяти выполнить свопинг.

```
vm.v_inactive_target: 3769
установка: runtime
```

Количество страниц памяти, которые система виртуальной памяти пытается очистить, когда оказывается заполненной.

```
vm.v_cache_min: 2513
установка: runtime
```

Минимальный желательный размер очереди кэша виртуальной памяти.

```
vm.v_cache_max: 5026
установка: runtime
```

Максимальный желательный размер очереди кэша виртуальной памяти.

```
vm.swap_enabled: 1
установка: /boot/loader.conf
```

Управляет использованием пространства свопа. Если значение равно 0, то система не будет выполнять свопинг. Если диск свопа поврежден или вы запускаете -current, можете попробовать применить это значение.

```
vm.swap_idle_enabled: 0
vm.swap_idle_threshold1: 2
```

```
vm.swap_idle_threshold2: 10
установка: runtime
```

Если в большой системе постоянно происходит свопинг, установка параметра swap_idle_enabled предпишет менеджеру виртуальной памяти помещать бездействующие процессы в виртуальную память быстрее, чем другие процессы. Параметр threshold предписывает системе, сколько секунд надо ждать, прежде чем отнести процесс к бездействующим. Вероятно, значения по умолчанию приемлемы; простое включение vm.swap_idle_enabled окажет свое влияние. Включайте этот параметр лишь в том случае, если виртуальная память в системе существенно нагружена!

```
vfs.vmiodirenable: 1
установка: runtime
```

Позволяет FFS использовать систему виртуальной памяти для кэширования поиска в каталогах, что позволит увеличить производительность дисков. В сочетании с кодом хеширования каталогов и soft updates такая функция улучшает доступ к диску на 6000%.

```
vfs.usermount: 0
установка: runtime
```

Если установлен этот параметр, пользователи могут монтировать файловые системы. И за счет этого применять дискеты и компакт-диски. Пользователь должен владеть точкой монтирования.

```
net.inet.ip.forwarding: 0
установка: runtime
```

Управляет способностью ядра перенаправлять пакеты. При наличии нескольких сетевых карт может потребоваться, чтобы система FreeBSD действовала как шлюз, маршрутизатор или брандмауэр. Если это значение установлено, система сможет осуществлять внутреннее перенаправление пакетов. При желании перенаправление можно включить или отключить.

```
net.inet.ip.redirect: 1
установка: runtime
```

Управляет способностью посылать пакеты переадресации[1] ICMP, если система предоставляет сервис маршрутизации. Этот параметр не оказывает эффекта, если система не выполняет маршрутизацию.

```
net.inet.ip.ttl: 64
установка: runtime
```

Максимальное количество «прыжков» (hops), которые любой протокол не-ICMP поддерживает в сети.

```
net.inet.ip.sourceroute: 0
установка: runtime
```

---

[1] Это требуется для управления маршрутизацией, если в сети используется динамическая маршрутизация. – *Примеч. науч. ред.*

Включает и выключает перенаправление пакетов от источника.

```
net.inet.ip.accept_sourceroute: 0
установка: runtime
```

Если значение параметра равно 1, то система принимает пакеты, маршрутизируемые в зависимости от их источника (source-routed packets),[1] предназначенные для нее. Если вы не знаете, что такое маршрутизация от источника, то поверьте мне на слово, что лучше обойтись без нее.

```
net.inet.ip.fastforwarding: 0
установка: runtime
```

Если вы предоставляете сервис маршрутизации, данный sysctl значительно улучшит пропускную способность канала. Это достигается за счет устранения проверок на наличие тривиальных ошибок в пакетах, а также за счет обхода правил фильтрации пакетов.

```
net.inet.icmp.drop_redirect: 0
установка: runtime
```

При значении 1 система будет игнорировать пакеты переадресации ICMP. Это не находит широкого распространения в общедоступном Интернете и лишь изредка применяется в частных сетях.

```
net.inet.icmp.log_redirect: 0
установка: runtime
```

При нормальных обстоятельствах система никогда не увидит переадресацию ICMP. Хотя такая возможность нужна в административных целях, об ее использовании вы узнаете. Включение этого sysctl заставляет систему протоколировать любые переадресации пакетов ICMP, которые она получает.

```
net.inet.icmp.bmcastecho: 1
установка: runtime
```

Когда установлен этот параметр, система будет отвечать за запросы ICMP по широковещательному адресу (адрес с самым большим номером в блоке IP-адресов). Это необходимо для соответствия стандартам, однако такая возможность была источником неприятностей, поэтому по умолчанию она отключена.

```
net.inet.tcp.rfc1323: 1
установка: runtime
```

Включает алгоритмы масштабирования окна, описанные в RFC 1323.

```
net.inet.tcp.rfc1644: 0
установка: runtime
```

Включает Transactional TCP, как описано в RFC 1644.

---

[1] source-routing – это возможность маршрутизировать пакеты от разных источников через заранее определенные пути. – *Примеч. науч. ред.*

```
net.inet.tcp.sendspace: 16384
net.inet.tcp.recvspace: 16384
установка: runtime
```

Количество разрядов, зарезервированных для буферов приема и отправки. Когда бы ни было открыто соединение, система резервирует для него буферы приема и отправки. По умолчанию эти значения равны 16 384, или 16 Кбайт. Если в системе мало нагруженных соединений, эти значения можно увеличить. Значение 32 768 будет подходящим. Не изменяйте этот sysctl, если система обслуживает большое количество соединений, – вы значительно увеличите нагрузку на систему и нанесете удар производительности. Эти значения играют важную роль в вычислении параметра памяти ядра NMBCLUSTERS; их увеличение повлечет за собой увеличение объема памяти, выделяемого для mbufs. Стоит только излишне увеличить эти значения, как ядро впадет в панику во время начальной загрузки.

```
net.inet.tcp.log_in_vain: 0
установка: runtime
```

Протоколирует попытки подсоединения к любому порту TCP, который не прослушивает ни одна программа.

```
net.inet.tcp.blackhole: 0
установка: runtime
```

По умолчанию TCP/IP возвращает код ошибки, когда пытается подключиться к закрытому порту. Это проявляется как ошибка «connection reset by peer». Если присвоить этому параметру значение 1, то попытки подключения к порту TCP блокируются, но сообщение об ошибке не посылается. Это замедляет сканирование портов и может улучшить кажущуюся безопасность вашей системы. Однако это не замена фильтрации пакетов!

```
net.inet.tcp.delayed_ack: 1
установка: runtime
```

Предписывает системе включать в пакеты данных информацию о демонтаже соединения TCP, а не посылать дополнительные пакеты, сигнализирующие о завершении соединения.

```
net.inet.tcp.path_mtu_discovery: 1
установка: runtime
```

Включает обнаружение минимального MTU на пути пакета (Path MTU).[1]

```
net.inet.tcp.slowstart_flightsize: 1
установка: runtime
```

Задает количество пакетов, которые можно послать во время медленного старта транзакции TCP в глобальной сети.

---

[1] Более детально с Path MTU можно ознакомиться в RFC 1812, например по адресу *http://www.ysn.ru/doc/cie/RFC/1812/62.htm*. – *Примеч. науч. ред.*

```
net.inet.tcp.local_slowstart_flightsize: 65535
установка: runtime
```

Задает количество пакетов, которые можно послать во время медленного старта транзакции TCP в локальной сети.

```
net.inet.tcp.newreno: 1
установка: runtime
```

Включает и выключает восстановление соединений согласно RFC 2582. Эта функция известна как TCP NewReno Algorithm.

```
net.inet.tcp.do_tcpdrain: 1
установка: runtime
```

Предписывает системе выбрасывать пакеты из повторно собранной очереди, когда в системе ощущается нехватка mbufs.

```
net.inet.tcp.always_keepalive: 1
установка: runtime
```

Если установлено значение 1, будут найдены и уничтожены старые соединения. Это ненамного увеличит объем сетевого трафика, но очистит сервер, работающий без останова 30 месяцев, от всего лишнего, что в нем накопилось за это время. Если установлено значение 0, то останутся действующими даже ненадежные соединения. Это компромисс между долговременной стабильностью и кратковременным удобством.

```
net.inet.udp.log_in_vain: 0
установка: runtime
```

Протоколирование попыток соединения с любыми портами UDP, которые не прослушивает ни одна программа.

```
net.inet.udp.blackhole: 0
установка: runtime
```

По умолчанию TCP/IP возвращает код ошибки при попытке подсоединения к закрытому порту. Это проявляется как ошибка «connection reset by peer». Если присвоить этому параметру значение 1, то попытки подключения к порту UDP блокируются, но сообщение об ошибке не посылается. Это замедляет сканирование портов и может улучшить кажущуюся безопасность вашей системы. Но фильтрацию пакетов не заменит!

```
hw.ata.ata_dma
установка: /boot/loader.conf
```

Управляет использованием DMA в устройствах IDE. Это современный стандарт. Установите значение 0, если ваши аппаратные средства вместо DMA используют PIO. (Если у вас есть аппаратные средства PIO, то вы, вероятно, об этом знаете.)

```
hw.ata.wc
установка: /boot/loader.conf
```

Управляет кэшированием записи в дисках IDE. Значение 1 увеличит производительность ценой целостности данных в случае аварии системы.

```
hw.ata.tags: 0
установка: /boot/loader.conf
```

Включает организацию упорядоченных очередей.[1] Такую возможность поддерживают только определенные жесткие диски IBM. Если у вас есть такие диски, то эта возможность ясно указана на упаковке.

```
hw.ata.atapi_dma: 0
установка: /boot/loader.conf
```

Управляет моделью доступа DMA в устройствах ATAPI. Обратитесь к документации по аппаратным средствам и посмотрите, поддерживают ли они DMA. В ATAPI могут быть трудности с DMA, поэтому по умолчанию данная возможность отключена. Ее установка легко может подвесить систему.

```
jail.set_hostname_allowed: 1
установка: runtime
```

Определяет, может ли владелец клетки изменить имя хоста своей клетки. См. главу 8.

```
jail.socket_unixiproute_only: 1
установка: runtime
```

Определяет, может ли владелец клетки использовать протоколы, отличные от TCP/IP. См. главу 8.

```
jail.sysvipc_allowed: 0
```

Определяет, может ли владелец клетки использовать вызовы System V IPC calls. См. главу 8.

```
compat.linux
установка: read-only
```

Эти sysctls предоставляют информацию для модуля совместимости с Linux. См. главу 11.

---

[1] Упорядоченные очереди (tagged queueing) позволяют устройству обрабатывать несколько транзакций одновременно, часто переупорядочивая их для уменьшения числа и расстояния перемещений головок жесткого диска. Применяется в некоторых жестких дисках, обычно в SCSI. – *Примеч. науч. ред.*

Алфавитный указатель

Числа

–1, сигнал, 556

A

% a, переменная, 205
–a, ключ, 289, 382
ABI (Application Binary Interface), 314, 318
additional, раздел в dig, 333
alert, уровень, 507
ALL и ALL EXCEPT, ключевые слова, 202
Allow, оператор, 415
allow, параметр, 202
AllowGroups, параметр, 375
AllowOverride None, строка, 414
AllowOverride, параметр, 417
AllowTcpForwarding, параметр, 375
AllowUsers, параметр, 375
Angband, игра, 580
answer, раздел в dig, 332
aout, тип исполняемых файлов, 311
Apache, веб-сервер, 405
  каталоги, защищаемые с помощью паролей, 421
  конфигурирование раздела Directory, 413
  управление, 422
  управление HTTP-запросами, 419
apache13, версия веб-сервера, 405
apache13+ipv6, версия веб-сервера, 405
apache13-fp, версия веб-сервера, 406
apache13-modssl, версия веб-сервера, 406
apache13-ssl, версия веб-сервера, 406
apache2, версия веб-сервера, 406
apachectl, команда, 422
apache-jserv, версия веб-сервера, 405
APOP (Authentication POP), протокол, 403
apropos, команда, 62
ARP (Address Resolution Protocol), 126, 132
ATM (Asynchronous Transfer Mode), 126, 264
auth, средство, 505
AuthName, директива, 420
authority, раздел в dig, 333
authpriv, средство, 506
avm, колонка, 495

B

–b, ключ, 143
b, колонка, 495
bash, командная оболочка, 54
Berkley Packet Filter, 115
BGL (Big Giant Lock), блокировка, 325
BIND
  in-addr.arpa, 341
  прямые и обратные DNS, 339
blanktime, параметр, 271
BSD Today, веб-сайт, 64
BSD4.3, функции, 112

C

% c, переменная, 205
–c (создать новый архив), режим, 77
–c, ключ, 382
CD-проигрыватели, программы, 578
cfgmaker, инструмент, 526
CFLAGS= –0 –pipe, параметр, 250
CHANGE_THIS.FreeBSD.org, 157
CheckHostIP, параметр, 373
check-in (ci), команда, 86
check-out (co), команда, 86
chkgrp, команда, 179

chmod (change mode), 183
chroot (changed root), 219
CIFS (Common Internet File Sharing), 564–565
Cipher blowfish, параметр, 374
Ciphers, параметр, 375
Civilization, игра, 580
clear_tmp_enable, параметр, 273
clear-text-password, параметр, 402
COMPATxx=yes, параметр, 254
+COMMENT, файл, 288
+CONTENTS, файл, 288
configtest, ключ, 422
console, средство, 506
COPTFLAGS= –0 –pipe, параметр, 250
copyright, свойство, 247
coredumpsize, переменная, 244
count, поле, 512
cpu, оператор, 109
cputime, переменная, 244
crit, уровень, 507
cron, средство, 506
crontab, файл, 233
cs, колонка, 496
csh.*, файлы, 235
-current, 148
cvpasswd, команда, 171
CVS (Concurrent Versions System), 153, 251
cvs-supfile, 156
CVSup
  ID, 170
  локальный сервер, 166
  обновление FreeBSD
    блокирование обновлений CVSup с помощью файла refuse, 159
    выбор supfile, 155
    записи устройств, 164
    модификация supfile, 157
    обновление исходного кода системы, 159
    обновление ядра, 161
    сборка новой системы FreeBSD, 160
    указание нескольких коллекций, 158
    упрощение процесса, 166
    установка системы, 162
cvsup17.FreeBSD.org, 157
cvsupd, 492

D

%d, переменная, 205
–d (поиск различий в файлах), ключ, 78
–D, операторы, 424
Daemon News, 30
daemon, группа, 180
dangerously dedicated mode, особо опасный режим, 38
data, команда, 385
datasize, переменная, 244
dbmmanage, утилита, 420
DDB_UNATTENDED, параметр, 540
DDP (Datagram Delivery Protocol), 139
death spiral, см. спираль смерти
debug, корень дерева sysctl MIB, 99
debug, уровень, 508
default, класс, 243
Defaultpasswd, запись, 231
defaultrouter="", 271
DenyGroups, параметр, 375
/dev, изменения, 164
/dev/MAKEDEV, сценарий, 165
dhclient.conf, файл, 235
DHCP, конфигурирование, 52
  клиента DHCP, 235
dialer, группа, 180
diff, формат, 164
dig, команда, 333
Digital UNIX (OSF/1), 318
Directory, раздел,конфигурирование, 413
dmesg, файл, 459
dmesg.boot, файл, 72, 106, 109
DNS
  запуск во время начальной загрузки, 354
  защита, 355
  изменения, 353
  источники информации, 344
  как он работает, 329–330
  конфигурирование клиента, 337
  основные инструменты, 335
  управление порядком предоставления информациии, 356
doc-all, коллекция, 158
doc-supfile, 156
domain, ключевое слово, 336
Doom, игра, 580
dotdir, запись, 231
drive, ключевое слово, 472

dummynet, инструмент, 358
dump, поле, 440
dump, программа, 440, 543–551
gdb, 547
vmcore и безопасность, 548
изучение переменных, 546
изучение строк, 546
отладка без символов, 549
расширенная отладка ядра, 545
dumpdev, местоположение, 541
dumpdev="NO", 273
dumpon, команда, 541

E

ELF, тип исполняемых файлов, 311
Emacs, текстовый редактор, 54
emerg, уровень, 507
Enable, пункт меню настройки, 51
ENABLE_SUIDPERL=true, параметр, 251
err, уровень, 507
ErrorLog, директива, 411
/etc, дерево, 222
/etc, изменения, 162–164
/etc, каталог, 229–275
/etc/adduser.conf, 233
/etc/csh.*, файлы, 235
/etc/defaults/rc.conf, 230
/etc/dhclient.conf, файл, 235
/etc/fstab, файл, 237, 439, 454
/etc/ftp.*, файлы, 237
/etc/hosts.allow, файл, 237
/etc/hosts.equiv, файл, 238
/etc/hosts.lpd, файл, 239
/etc/locate.rc, файл, 239
/etc/login.access, файл, 240
/etc/login.conf, файл, 242
/etc/mail/mailer.conf, файл, 248
/etc/make.conf и /etc/defaults/make.conf, файлы, 249
/etc/master.passwd, 255
/etc/motd, файл, 257
/etc/mtree/*, файлы, 258
/etc/namedb/*, файлы, 258
/etc/newsyslog.conf, файл, 258
/etc/passwd, файл, 258
/etc/periodic.conf и /etc/defaults/periodic.conf, 259
/etc/profile, файл, 262
/etc/protocols, файл, 263
/etc/pwd.db, файл, 264
/etc/rc.atm, сценарий, 264
/etc/rc.conf и /etc/defaults/rc.conf, файлы, параметры коммутируемого соединения PPP, 269
/etc/firewall, сценарий, 265
/etc/ftpchroot, файл, 434
/etc/inetd.conf, 379–380
/etc/printcap, файл, 260, 571
/etc/rc.diskless1 и .diskless2, сценарии, 264
/etc/rc.firewall6, сценарий, 265
/etc/rc.i386, сценарий, 265
/etc/rc.isdn, сценарий, 265
/etc/rc.network, сценарий, 265
/etc/rc.network6, сценарий, 265
/etc/rc.serial, сценарий, 265
/etc/rc.syscons, сценарий, 265
/etc/rc.sysctl, сценарий, 265
/etc/security, файл, 274
/etc/services, файл, 274
/etc/shells, файл, 275
/etc/spwd.db, файл, 275
/etc/ssh, файл, 275
/etc/ssh/ssh_host_dsa_key, 370
/etc/ssh/ssh_host_key, файл, 374
/etc/ssh/ssh_host_key.pub, файл, 374
/etc/sysctl.conf, файл, 275
Ethernet, 146
IP-адреса, 136
UDP и TCP, 137
сетевые порты, 138
хабы и коммутаторы, 133
ExecCGI, параметр, 416
ext2fs, файловая система, 453

F

–f, ключ, 78, 81
FallBackToRsh no, параметр, 373
FAQ, 59, 62
FCODES, переменная, 239
fdimage.exe, программа, 34
filesize, переменная, 244
flt, колонка, 495
FollowSymLinks, параметр, 416
form feed upon open, параметр печати, 261
Forth, язык программирования, 104
ForwardX11 yes, параметр, 373
fre, колонка, 495

FreeBSD 3.0 SMP, 325
FreeBSD 5 SMP, 326
FreeBSD Mall, веб-сайт, 30
FreeBSD, веб-сайт, 30
FreeBSD, обновление
  с компакт-диска, 17
  с помощью sysinstall, 152
FreeBSD, сообщество, 582
FreeBSD-current, 148
FreeBSD-hackers, почтовые рассылки, 38
fsck, инструмент, 447
fstab, файл, 237
FTP (File Transfer Protocol), 430
  установка CVSup по, 154, 155
  установка FreeBSD по, 32
  установка пакетов по, 286
ftp, средство, 506
ftp.*, файлы, 237
ftpd, программа, 199
fullstatus, ключ, 422

G

games, группа, 180
gateway_enable="NO", параметр, 271
gdb, отладчик, 543
GENERIC, ядро, 106, 161
GID, уникальный цифровой идентификатор группы, 179
gnats-supfile, 156
GNU tar, 76
GnuCash, программа, менеджер денежных операций, 579
grep, команда, 194, 282

H

%h, переменная, 205
–h, флаг, 81
HAVE_MOTIF=yes, параметр, 255
HEADS UP, сообщение, 150
Heretic, игра, 580
host, команда, 330
host.allow, параметр, 247
host.deny, параметр, 247
hostname, параметр, 267
HostnameLookups, оператор, 411
/hosts.allow, файл, 247
/hosts.equiv, файл, 238
/hosts.lpd, файл, 239
HTTP (Hypertext Transfer Protocol), 405
httpd, программа, 199
httpd, серверы, 408
hushlogin, переменная, 245
hw, корень, 99

I

–i 'pattern', параметр, 160
–I, ключ, 142
i386 (тип процессора), 250
i486 (тип процессора), 250
i586 (тип процессора), 250
i586/mmx (тип процессора), 250
i686 (тип процессора), 250
ibcs2_enable="NO", 273
ICMP (Internet Control Message Protocol), 127
  переадресация, 590
icmp_drop_redirect="NO", 268
icmp_log_redirect="NO", 268
id, команда, 178
ident, оператор, 110
ifconfig_lo0="inet 127.0.0.1", 268
ignorenologin, переменная, 245
IgnoreRhosts yes, параметр, 375
IN, класс, 332
in, колонка, 496
Include, оператор, 413
Includes, параметр, 416
IncludesNOEXEC, параметр, 416
INDEX, файл, 286
Indexes, параметр, 417
inetd, демон, 177, 195, 382
inetd_enable="NO", 270
info, уровень, 508
init, программа, 588
inodes, индексные дескрипторы, 442
INSTALL=install –C, параметр, 250
IP (Internet Protocol), 127
IPC, межпроцессное взаимодействие, 99
ipf, команда, 215
IPFilter, 208
  протоколирование, 218
  реализация, 208
IPFILTER_LOG, параметр, 208
IPFW, инструмент фильтрации пакетов, 209
ipmon, программа, 218

IPv6, 380
конфигурирование, 52
поддержка, 127
сетевая конфигурация, 265
сокеты, 193
IP-адрес, 127
сетевые маски, 136
совмещение аутентификации и доступа, 172
управление доступом, 169, 414
ISDN (Integrated Services Digital Network), 126
ISO 8601, формат времени, 513

J

jail.socket_unixiproute_only, sysctl, 221
jail.sysvipc_allowed, sysctl, 222
JVM (Java virtual machine), 372

K

k5 (тип процессора), 250
k6 (тип процессора), 250
k6-2 (тип процессора), 250
k7 (тип процессора), 250
KeepAlive, параметр, 375
kern, корень дерева sysctl MIB, 99
kern, средство, 506
kern.hostname, 98
kern_securelevel="–1", параметр, 274
kern_securelevel_enable="NO", параметр, 274
kernel.debug, файл, 540
key, параметр поиска, 283
keymap="NO", параметр, 271
kill –15 –1, команда, 227
killall snmpd, команда, 310
kldload, инструмент, 105
kmem, группа, 180
KNOWN, ключевое слово, 201
koffice, офисный пакет, 578

L

–l, ключ, 382
LD_LIBRARY_PATH, переменная окружения, 313
ldconfig_paths="/usr/lib/compat /usr/X11R6/lib /usr/local/lib", параметр, 273
LINT, конфигурация ядра, 119, 160
linux_enable="NO", параметр, 273
Listen, оператор, 410
load, команда, 104
loader.conf, 101
LOCAL, местоположение источника подключения, 242
LOCAL, ключевое слово, 200
local0 – local7, средства, 507
locate.rc, файл, 239
log_in_vain="NO", параметр, 267
log-file turned over, сообщение, 514
Login ID, 49
login.access, файл, 240
lpd (line printer daemon), 572
lpr, средство, 506
ls, команда, 104
lsof, инструмент, 228

M

machdep, корень дерева sysctl MIB, 99
machine, оператор, 109
MAC-адрес (Media Access Control), 126
magic, файл, 407
Mahjongg, игра, 580
mail, группа, 180
mail, средство, 506
mailer.conf, файл, 248
make buildkernel, команда, 161
make clean, команда, 303
make configure, этап make install, 296
make depend && make all install, команда, 116
make depends, этап make install, 295
make extract, этап make install, 296
make fetch, этап make install, 295
make package, команда, 301
make patch, этап make install, 296
make reinstall, команда, 299
make update, команда, 166
make world, 152
make checksum, этап make install, 295
make.conf, файлы, 249
man –k, команда, 62
man, группа, 180
manpath, переменная, 245
mark, средство, 506
master.cf, файл, 390
master.passwd, 255
maxproc, переменная, 244

maxusers, значение, 110
MAXUSERS, параметр ядра, 121
MBR (Master Boot Record), 38
mbufs, 121–123, 129
md, псевдоустройство 115
memoryuse, переменная, 244
mergemaster, утилита, 162
mfsroot.flp, файл, 34
MIB (Management Informantion Base), 98, 100, 516
Microsoft Word, 284
Minesweeper, игра, 580
minpasswordlen, свойство, 246
mixpasswordcase, свойство, 247
MODULES_WITH_WORLD=true, параметр, 254
motd, файл, 257
mount –a, команда, 93
mount_smbfs, ключи, 567
moused_enable="NO", параметр, 272
moused_flags="", параметр, 272
moused_port="/dev/psm0", 272
moused_type="", 272
Mozilla, веб-броузер, 576
mput, команда, 432
MRTG, 525
  индексная страница, 533
  конфигурирование, 526
  мониторинг отдельного MIB, 531
  настройка, 531
  отслеживание других системных значений, 529
  полезные MIB, относящиеся к net-snmp, 530
  тестирование, 528
MS-DOS, файловая система, 455
mt, команда, 75
mtree/*, файлы, 258
MultiViews, параметр, 417
mutex, 326
$myhostname, переменная, 390

N

%n, переменная, 206
–n, ключ, 142
named
  in-addr.arpa, 341
  первичный и вторичный серверы DNS, 339
  прямые и обратные DNS, 339
named, демон, 220, 345
  конфигурирование, 344
named.conf, файл, 341, 344
named_enable="NO", 270
named_enable="YES", 354
named_flags="–u bind –g bind", 270
named_program="named", 270
namedb/*, файлы, 258
NAT (Network Address Translation), 337
ndc (name daemon controller), 353
nessus, инструмент, 228
net, корень дерева sysctl MIB, 99
net.inet.tcp.recvspace, приемный буфер, 122
Nethack, игра, 580
Netscape, веб-броузер, 575
net-snmp, пакет, 517
netstat –m, команда, 144
netstat –nr, команда, 145
netstat –r, команда, 144
netstat, команда, 122, 141
Network Configuration, пункт меню, 53
network_interfaces="auto", параметр, 268
New Xfree86, 54
news, группа, 180
news, средство, 506
newsyslog.conf, файл, 258
NICE, колонка, 488
niceness, изменение приоритетов с помощью nice, 491
NMBCLUSTERS, параметр ядра, 592
NO_BIND=true, параметр, 251
NO_CVS=true, параметр, 251
NO_FORTRAN=true, параметр, 251
NO_GAMES=true, параметр, 253
NO_LPR=true, параметр, 252
NO_MAILWRAPPER=true, параметр, 252
NO_MODULES=true, параметр, 252
NO_OBJC=true, параметр, 252
NO_OPENSSH=true, параметр, 252
NO_OPENSSL=true, параметр, 252
NO_SENDMAIL=true, параметр, 252
NO_SHAREDOCS=true, параметр, 252
NO_TCSH=true, параметр, 253
NO_X=true, параметр, 255
noatime, параметр, 445
noauto, поле, 440
nobody, группа, 180

NOCRYPT=true, параметр, 253
nodev, параметр, 445
nodump, флаг, 80
NOERROR, сообщение, 332
noexec, параметр, 445
nogroup, группа, 180
NOINFO=true, параметр, 253
NOLIBC_R=true, параметр, 253
nologin, переменная, 245
NOMANCOMPRESS=true, параметр, 254
none, уровень, 508
NOPERL=true, параметр, 253
NOPORTDOCS=true, параметр, 255
NOPROFILE=true, параметр, 254
norecurse, ключ, 335
nosuid, параметр, 445
nosymfollow, параметр, 445
notice, уровень, 508
NOUUCP=true, параметр, 254
nsmb.conf, файл
  ключевые слова, 565
  образцы записей, 569
NTP (Network Time Protocol), 377
ntp, средство, 506
ntpd, демон, 379
ntpdate, 378–379

O

Objective C, поддержка, 252
OpenBSD, 173
openfiles, переменная, 244
OpenOffice, офисный пакет, 577
openssl list-cipher-commands, команда, 403
operator, группа, 181
Order, оператор, 414
OSF/1 (Digital UNIX), 317
osf1_enable="NO", параметр, 273

P

p1003_1b, корень дерева sysctl MIB, 99
p2 (тип процессора), 250
p3 (тип процессора), 250
Pac Man, игра, 580
Package Selection, меню, 55
PACKAGEROOT, переменная окружения, 291
PACKAGESITE, переменная окружения, 291
PageTop, тег, 528
Palm, программы для, 579
PARANOID, ключевое слово, 201
Pass#, поле, 440
passwd, файл, 258
passwd_format, свойство, 246
PasswordAuthentication, параметр, 376
Path MTU, обнаружение, 592
path, запись, 232
path, переменная, 245
periodic.conf, файл, 259
permission denied, ошибка, 224
PermitEmptyPasswords no, параметр, 376
PermitRootLogin no, параметр, 376
pi, колонка, 495
PID, значения, 485
PID, колонка, 193, 488
Pine, почтовая программа, 577
pipe, правила, 361
pkg_add, инструмент, 286
pkg_delete –f, команда, 306
pkg_delete, команда, 289, 299
pkg_info, команда, 288
PKG_TMPDIR, переменная окружения, 290
pkg_version, инструмент, 304
po, колонка, 495
POP3, 403
  qpopper, 399
  конфигурирование pop3ssl, 403
  конфигурирование qpopper по умолчанию, 401
  протоколирование, 397
  режимы, 397
  тестирование, 397
  установка, 396
pop3sssl, 398
Port 22, параметр, 374
ports-supfile, 156
POSIX, 99, 113
POST, метод, 418
postconf, программа, 395
postfix, 388–391, 395
  postfix check, команда, 395
  postfix reload, команда, 395
  postfix start, команда, 395
  postfix stop, команда, 395
PPP (Point-to-Point Protocol), протокол точка-точка, 114
ppp_enable="NO", параметр, 269

ppp_mode="auto", параметр, 269
ppp_nat="YES", параметр, 269
PPP_NOSUID=true, параметр, 251
ppp_profile=, 269
PR (Problem Report), отчет о проблеме, 557–562
PREFIX, переменная, 301
printcap, файл, 260, 571
priority, переменная, 246
procfs, файловая система процессов, 112, 223, 226
proto, ключевое слово, 212
Protocol 2,1, параметр, 374
ps –ax, инструмент, 226
pwd.db, файл, 264

Q

qmgr (queue manager), 389
qpopauth, команда, 402
qpopper, безопасность, 403
Quake, игра, 580
query, раздел в dig, 332

R

–r, ключ, 144, 434
r, колонка, 495
RAID
  выделение разделов для Vinum, 471
  замена поврежденного зеркалированного плекса, 481
  запуск Vinum при начальной загрузке, 477
  зеркалированные тома, 476
  команды Vinum, 477
  конфигурирование Vinum, 471
  объединенный плекс, 468
  подготовка накопителей Vinum, 470
  сравнение аппаратных и программных реализаций, 465
  типы плексов Vinum, 468
  тома, разбитые на блоки, 475
  удаление конфигурации Vinum, 475
  уровни, 466
RAM, 31
rc.conf, файл, 230, 266
  параметры сетевой маршрутизации, 271
rc.shutdown, сценарий, 274
rcs, команда, 86
rcsdiff, команда, 91
re, колонка, 495
reboot, оператор, 236
refuse, файл, 159
reject, оператор, 237
relayhost, ключевое слово, 392
-release, 150, 151, 152
requirehome, переменная, 246
resetconfig, команда, 474
retr, команда, 397
RFC 2582, восстановление соединений, 593
RhostsAuthentication yes, параметр, 373
ro, параметр, 440
Rotor, кодовое название SSCLI, 429
route, команда, 140
router_enable="NO", параметр, 271
rp, запись, 571
RPM (Red Hat Package Manager), 323
rtld (run-time linker), 311
runtime dependencies, см. зависимости запуска

S

–s, ключ, 219
sa, устройство, 71
Samba, 564
sappnd, флаг, 186
savecore, программа, 541
/sbin, команды, 94
sbsize, переменная, 244
schg, флаг, 187
SCO UNIX, 318
SCSI, 460
  жесткие диски, 31
search, ключевое слово, 336
Secure Connectionless Internet Protocol, 293
security, средство, 506
security, файл, 274
send, оператор, 237
send_message, запись, 231
sendmail, демон, 220, 248, 388
sendmail_outbound_enable="YES", параметр, 273
ServerName, оператор, 411
set, команда, 104
setenv, переменная, 246
$SHELL, переменная окружения, 246

shells, файл, 275
shutdown now, команда, 162
shutdown, программа, 227
SIZE, колонка, 488
SLIP (Serial Line Internet Protocol), 114
SMB, инструменты, 565
SMP (Symmetric MultiProcessing), 323–328
SMTP (Simple Mail Transfer Protocol), 138
SNMP, 515
  MIB (Management Informantion Base), 516
  net-snmp, 517
  snmpwalk, 518
  основы, 516
  преобразование номеров и имен, 519
  специальные запросы snmpwalk, 518
SNMP for the Public Community, веб-сайт, 535
snmpwalk, 518
sockstat, инструмент, 193
Soft Updates, 44, 450
spwd.db, файл, 275
src/crypto, каталог, 254
src/share, каталог, 254
src-all, тег, 157
SSH (Secure Shell), 369
  включение, 370
  конфигурирование, 376
  основы, 371
  соединение через, 372
  тестирование, 370
ssh hostname, команда, 372
ssh, файл, 275
sshd, демон, 221
sshd_enable="NO", параметр, 270
sshd_flags="", параметр, 271
SSL (Secure Sockets Layer)
  FrontPage и, 425
stable-supfile, 156
stacksize, переменная, 244
staff, группа, 181
standard-supfile, 156
StarOffice, офисный пакет, 578
start, ключ, 422
start_vinum="NO", параметр, 274
StartServers, значение, 408
startssl, ключ, 422
status, ключ, 422
status, файл, 226
stop, ключ, 422
StrictHostKeyChecking no, параметр, 374
sunlnk, флаг, 187
supfile, 155
SVR4 (System V Release 4), 318
svr4_enable="NO", параметр, 273
sy, колонка, 496
SymLinksIfOwnerMatch, параметр, 417
SYN, флаг, 213
SYN-ACK, 211
SYN-ACK, флаг, 213
SYN+ACK, состояние, 143
SYN_RCVD, состояние, 143
syscontact, 521
sysctl, программа, 97
sysctl.conf, файл, 275
sysinstall, 37, 460
syslog, средство, 506
syslog.conf, файл, 508
syslogd_enable="YES", параметр, 269
syslogd_flags="–s", параметр, 270

T

–t (перечислить все файлы архива), режим, 77
–t, ключ, 453
tag=., ключевое слово, 158
TAPE, переменная, 74
tar, программа, 75–79
tarballs, 79
Target, ключевое слово, 527
TCP (Transmission Control Protocol), 593
TCP Wrappers, 199–207
  allow, параметры, 202
  изменение работы inetd, 382
  имя демона, 199
  параметры для ответных действий, 206
  примеры, 207
tcp_drop_synfin="NO", параметр, 267
tcp_extensions="NO", параметр, 267
teletype (tty), 556
telnet, 369
term, переменная, 246
times.allow, свойство, 247
times.deny, свойство, 247

timezone, переменная, 246
tip, программа, 554
Title, тег, 528
TMPDIR, временный системный каталог, 239
TMPDIR, переменная окружения, 290
toggles, см. «переключатели»
TransferLog, директива, 428
trap, см. ловушка
tty, группа, 181
typescript, файл, 86
$TZ, переменная окружения, 246

U

–u, ключ, 82
uappnd, флаг, 187
uchg, флаг, 187
UCONSOLE, параметр, 112
UDP (User Datagramm Protocol), 137
UFS (UNIX File System), 116
UID (user ID), 49
umask, переменная, 246
UNIX-to-UNIX Copy Protocol (UUCP), 254
UNKNOWN, ключевое слово, 200
unload, команда, 104
usbd_enable="NO", параметр, 272
/usr/obj, файл, 501
/usr/src/contrib, файл, 562
/usr/src/UPDATING, файл, 160
UseLogin, параметр, 376
user, корень дерева sysctl MIB, 99
user, команда, 397
User, параметр, 410
user, средство, 507
USERNAME, колонка, 488
uucp, средство, 507
uunlnk, флаг, 187

V

/var, раздел, 43
/var/named, каталог, 342
/var/named/master/absolutebsd.com, файл, 351
vcount, функция, 545
verbose, запись, 231
vfs, корень дерева sysctl MIB, 99
vfs.usermount, sysctl, 101
vfs.vmiodirenable, параметр, 451
vi, текстовый редактор, 56
Vinum
  дисковые компоненты, 467
  запуск при загрузке, 477
  команды, 477
  объединенный плекс, 468, 471
vinum list –r, команда, 478
vinum printconfig, команда, 478
vinum.conf, файл, 471
vipw, программа, 256
vm, корень дерева sysctl MIB, 99
vmcore, файл, 548
vmstat, программа, 497
vnodes, виртуальные индексные дескрипторы, 443

W

–w, ключ, 101, 382
w, колонка, 495
warning, уровень, 508
welcome, переменная, 246
wheel, группа, 178, 181
WITH_FLEXRESP, запись, 298
WithPeak, параметр, 532
WorkDir, директива, 526
www-supfile, 156

X

–x (извлечь все файлы из архива), режим, 78
X11Forwarding, параметр, 376
X11R6, графический интерфейс, 573
Xevil, игра, 579
XFree86, графический интерфейс, 573
xterm, терминальное окно, 575

Z

–z (gzip), ключ, 78
–z, ключ, 541

А

абстракция интерфейса, 129
аварии и паника в системе, 562
  ответные действия, 542
  представление отчета о проблеме, 562
  применение dump, 543–551
    gdb, 543, 547
    vmcore и безопасность, 548

изучение переменных, 546
изучение строк, 546
нужны ли символы, 549
расширенная отладка ядра, 545
результаты, 548
что вызывает панику, 536
автоматическая перезагрузка, 540
адрес порта памяти, 36
адресная запись, 332
активные файлы, перенос, 463
алгоритмы масштабирования окна, 591
анонимные серверы FTP, 436
аппаратные последовательные консоли, 551
аппаратные средства
настройка, 34
распознавание, 106
асинхронное монтирование, 444
атака грубой силой, 402
атомные часы, 377
аутентификация, 170

Б

база данных с паролями, 264
бездействующий процесс, 487
библиотеки профилирования, 254
биты, 130
брандмауэр, 174
буферы, 174

В

веб-броузеры, 576
веб-сайт FreeBSD, 156
веб-серверы
Apache, 405
управление, 422
функции, указываемые в каталоге Directory, 418
механизм работы, 404
версии FreeBSD, 147
виртуальная память, 42, 451, 586
виртуальные домены, 395
виртуальные хосты
на основе IP, 427
на основе имен, 427
виртуальный хостинг, 429
временное монтирование, 463
время жизни (TTL), 346
время, задание, 234
время, поле, 513
встроенные функции портов, 296
вторичный DNS, 338
второй жесткий диск, 47
вход в систему
последовательный, 555
при чрезвычайных обстоятельствах, 556
выгрузка модулей, 104
вытесняющая многозадачность, 325

Г

графические форматы, 578
группы, 178
«грязные» диски, 446

Д

дамп, результаты применения, 548
двоичные операции, 130
демон SSH, включение, 271
дерево исходного кода FreeBSD, 155
диск восстановления, 94
дискеты, создание, 454
диски IDE, 31
разделы, 439
распределение пространства свопинга, 42
дисковый ввод/вывод, 484
добавление новых жестких дисков, 460
документация, FreeBSD, 62
домашний каталог пользователя, поле в /etc/master.passwd, 257
доменное имя, 53
доменные сокеты UNIX, 142
домены
виртуальные, 395
доступ ко второму домену, 569
доступ к разделяемым файловым ресурсам
mount_smbfs, ключи, 569
владение файлами в CIFS, 569
инструменты SMB, 565
конфигурирование CIFS, 565
наборы символов, 564
образцы записей nsmb.conf, 569
предварительная подготовка, 564
управление, 169
драйверы устройств, 108

Ж

жесткие диски
  разбиение, 39
  рекомендации, 31

З

зависимости запуска, 282
загрузка
  FreeBSD, 31
  задержка при, 458
  запасное ядро, 118
  конфигурирование во время, 266
  по FTP, 431, 432
  прерывание, 542
загрузчик, 104
замораживание кода, 149
записи о хосте, 352
записи устройств, 113
заплатки, 280
запрет FTP-доступа, 435
зашифрованный пароль,
  поле в /etc/master.passwd, 256
защита, 173, 197
  FTP, 431
  inetd, 381
  группы, 178
  для vmcore, 548
  изменение прав доступа, 182
  пароли на автономной системе, 569
  пароль root, 178
  установка профилей безопасности, 177
  установка уровней безопасности, 189
зеркалированный плекс, 468
зоны, файл named.conf, 342

И

игры, 579
идентификатор группы,
  поле в /etc/master.passwd, 257
идентификатор пользователя,
  поле в /etc/master.passwd, 257
идентификационная фраза, 366
изменение
  sysctl, 100
  приоритетов, 491
  пути установки, 301
  работы inetd, 382
имена хостов, 132, 238
имя демона, 199
имя машины, 346
имя пользователя,
  поле в /etc/master.passwd, 256
индексные номера, 525
интерфейс обратной петли, 114
история версий, 91
исходный код системы, 159

К

клетки, 219
  для пользователей, 434
  заключительная настройка, 224
  запуск, 225
  конфигурирование сервера для использования клеток, 220
  останов, 227
клиентские ключевые слова, 200
клиенты, SSH, 371
ключевые слова, поиск программного обеспечения по, 283
код доступа, 181
количество процессов, 486
коллекции, 157
коллекция портов, 280
командная оболочка пользователя,
  поле в /etc/master.passwd, 257
коммутаторы, 132
конфигурирование
  /etc/fstab, 462
  CIFS, 566
  IPFW, 360
  MRTG, 526
  ntpd, 379
  pop3ssl, 403
  postfix, 391
  ssh, 376
  TCP Wrappers, 199
  программной последовательной консоли, 553
  сервера для использования клеток, 220
  сетевых карт, 54
  ядра, 97–104
корневая зона, 342
корневые серверы имен, 341
кросс-компиляция, 278
круглые скобки, 347
кэш второго уровня, 120

кэш, создание, 312
кэширование
записи, 450, 594
каталогов, 501
в виртуальной памяти, 451

Л

ловушка, 523
локальные зоны, 343
локальные почтовые клиенты, 399
локальный сервер CVSup, 166

М

максимальные ограничения на ресурсы, 245
«малыши со скриптами», 174
маршрутизатор по умолчанию, 136
устанавливаемый при начальной загрузке, 140
маршрутизация, 136
межпроцессная коммуникация (IPC), 99
многопользовательский режим, 105
модули Apache, 410
моментальные копии -current, 150
мониторинг долгосрочный с помощью MRTG
конфигурирование MRTG, 526
мониторинг отдельного MIB, 531
мониторинг систем не-BSD, 535
образцы конфигураций MRTG, 533
отслеживание других системных значений, 529
полезные MIB, относящиеся к net-snmp, 530
монтирование
noasync, 444
неродных файловых систем, 451
разделяемого ресурса, 567
только для чтения, 443
мышь, 51, 272

Н

набор псевдонимов, 141
накопители на лентах, 72
настольная система FreeBSD, 580
X Window System, 573
доступ к разделяемым файловым ресурсам, 563
доступ к серверам печати, 570
доступ клиентов Windows к разделяемым файлам UNIX, 570
локальные принтеры, 572
настольные утилиты, 579
настройка
первичного сервера имен, 344
производительности на практике, 497
ядра, 105
недоставленная почта, 396
номер сети, 239
носители для инсталляции, 47
нулевой указатель, 547

О

обновление FreeBSD, 172
версии, 147
«порты», 306
через CVSup, 166
обратный DNS, 339
обслуживание почтового ящика, 401
общий ключ, 171
объединенный плекс, 468
операторы сравнения, 509
описания трафика, 360
определения классов, 243
оптоволокно, 131
останов клетки, 227
ответные действия, параметры для, 203
ответственная сторона, 346
отладка ядра, 540
отпечаток ключа, 371
отчеты с помощью SNMP, 515
MIB (Management Informantion Base), 516
net-snmp, 517
snmpwalk, 518
основы SNMP, 516
преобразование номеров и имен, 519
специальные запросы snmpwalk, 518

П

пакеты, 128, 293
деинсталляция, 289
информация о пакете, 290
перенаправление, 590

перенаправление от источника, 591
принудительная установка, 292
система «портов» и пакетов, 279
управление pkg_add, 290
установка с CD-ROM, 286
установка по FTP, 287
что устанавливает пакет, 288
пакеты обратной связи, 216
параметры среды по умолчанию, 245
пароли
root, 47
база паролей APOP, 401
на автономной системе, 569
первичный DNS, 338
передача файлов, 432
перезапуск, 55
«переключатели», 586
перекомпиляция, 316
переменные, изучение, 546
перемещение файлов, 464
/usr/obj, файл, 501
нескольких файлов по FTP, 432
перенос (портирование), 314
перенос активных файлов, 463
перепланирование, 490
печать в шестнадцатеричном виде, 550
плекс, 468
повторная установка портов, 300
повторное использование записей SOA, 349
поддержка Ethernet, 114
подколлекции, 156, 158
подсоединение
к сети Ethernet, 139
через SSH, 372
подтверждение подлинности в SSH, 371
поиск библиотеки, 312
пользователи
FTP, клетки/chroot, 434
добавление, 49, 402
новые, создание, 186
просмотр списка, 402
удаление, 402
помеченные очереди, 594
помощь, 69
страницы руководства, 62
веб-сайты, 63
документация FreeBSD, 62
почтовые рассылки, 57, 67
«порты», 281, 302
X Window System, 255
виртуальный хостинг и, 428
встроенные функции, 299
изменение пути установки, 301
очистка, 301
установка, 294
порты, Ethernet, 138
последовательные консоли, 551–557
вход в систему
последовательный, 555
при чрезвычайных обстоятельствах, 556
выключение, 557
изменение конфигурации, 553
применение, 553
программные, 552
почтовые рассылки, 58
FreeBSD-current, 148
FreeBSD-hackers, 38
архивы, 62
почтовые службы, 403
POP3, 396
конфигурирование pop3ssl, 403
обзор, 387
почтовые программы, 393, 577
postfix, 388, 391
sendmail, 387
псевдонимы, 393
почтовый адрес администратора, 411
права
на выполнение, 185
на запись, 185
на чтение, 185
правовые ограничения, программное обеспечение, 284
приветственные сообщения, 435
привязка устройств, 459
прикладной уровень, 127
принимать по умолчанию или отвергать, 209
принудительное монтирование «чтение/запись», 448
проверка
конфигурации, 478
состояния, 477
типа исполняемого файла, 317
файловой системы пользователя, 127
программное обеспечение, 54
поиск, 283

правовые ограничения, 284
проверка версий, 305
расширенное управление, 328
SMP, 328
установка и включение режима Linux, 323
сборка, 279
программные последовательные консоли, 553
Проект документирования (The Documentation Project), 582
производительность системы, 482, 535
дисковый ввод/вывод, 484
отчеты с помощью SNMP, 515
net-snmp, 517
snmpwalk, 518
индексные номера, 525
настройка snmpd, 520
основы SNMP, 516
преобразование номеров и имен, 519
специальные запросы snmpwalk, 518
письма о состоянии, 504
протоколирование с помощью syslogd, 505
syslog.conf, файл, 508
долгосрочный мониторинг с помощью MRTG, 526
индексная страница MRTG, 533
конфигурирование MRTG, 526
мониторинг отдельного MIB, 531
настройка MRTG, 532
отслеживание других системных значений, 530
ротация протоколов с помощью newsyslog.conf, 511
средства, 505
уровни, 507
процессор и память, 484
использование
top, 484
памяти, 489
пространства свопинга, 489
процессора, 490
ресурсы компьютера, 482
свопинг, 497
сетевая пропускная способность, 484
просмотр списка пользователей, 402
пространство свопинга, 42, 462, 489, 589
протоколирование
POP3, 397
обращений к ftpd, 434
по имени программы, 510
почты, 393
хост, 510
протокольные сообщения
CVS, 90
просмотр, 90, 510
процессор и память, 484
использование
top, 484
памяти, 489
пространства свопинга, 489
процессора, 490
псевдонимы, электронная почта, 393
псевдоустройства, 115
путь к pid-файлу, поле, 515
путь к корню документов, 408

Р

работа в сети, 124
сетевые уровни, 124
разбиение диска, 39
развилочная бомба, 485
разделы
MS-DOS, монтирование, 452
Vinum, 467
своп и паника в ядре, 539
создание, 461
разделяемый ресурс, монтирование, 567
размер блока, 45
размонтирование, 449
разработчики, версия FreeBSD для, 151
разрешение адресов, 132
расширенное управление программным обеспечением, 328
запуск программного обеспечения из чужой ОС, 319
совместно используемые библиотеки, 310
сценарии запуска и останова, 310
установка и включение режима Linux, 323
редактирование файлов ядра, 110
редакторы, текстовые, 56

режим Linux, 323
резервное копирование
и восстановление
программы для создания резервных копий, 75
dump/restore, 79
восстановление из архива, 82
диск восстановления, 79
резервное копирование системы, 71
управление версиями
получение старых версий, 89
строки Ident и ident, 91
рекомендации
жесткие диски, 31
память, 31
процессор, 31
ресурсы компьютера, 482
ретрансляция почты на главной машине, 392
руководство по FreeBSD (manpages), 61–62

С

свободная память, 487
свопинг, 493, 589
связанная память, 487
секретный ключ, 421
семафор, 326
серверы
FTP, 433
почтовые, 386
сертификаты, 364, 421
сетевая пропускная способность, 484
сетевой протокол, 381
сетевые mbufs, 123
сетевые порты, 138
сетевые сервисы, 382
SSH, 376
демон inetd, 379
системное время, 379
управление пропускной способностью, 362
сжатие страниц документации, 254
синхронизация, 588
синхронное монтирование, 444
системное время, 379
скрытые файлы, 303
снятие блокировок, 89
совместно используемые библиотеки
управление, 310
создание новых пользователей, 186
сокеты, 193
сообщество FreeBSD, 581
составное монтирование, 464
спираль смерти, 494
список
клиентов, 202
процессов, 488
серверов имен, 337
Справочник (Handbook), 57
средняя нагрузка, 485
срок действия пароля, поле в /etc/master.passwd, 257
страницы руководства, 61, 254
строки идентификации, 92
сценарии запуска и останова, 310
съемные носители и /etc/fstab, 454

Т

таблица интерфейсов, 516
текстовые броузеры, 576
текстовые редакторы, 56
темное оптоволокно, 131
тестирование, 498
MRTG, 528
SSH, 370
принтера, 572
типы
исполняемых файлов, 311
монтирования, 443
процессов, 486
точки, 352
тройное рукопожатие, 137

У

удаление пользователей, 402
управление
версиями
получение старых версий, 89
просмотр истории версий файла, 91
просмотр протокольных сообщений, 90
строки Ident и ident, 91
ретрансляцией, 387
трафиком, 198
уровень логического протокола, 127
уровень физического протокола, 126
уровни серверов времени, 377

установка
POP3, 396
postfix, 388
портов, 294
режима Linux, 320
установка FreeBSD, 30
sysinstall, 37
второй жесткий диск, 47
выбор профиля безопасности, 177
загрузка FreeBSD, 31, 34
конфигурирование сетевых карт, 54
конфигурирование ядра для поддержки карт ISA, 36
настройка после инсталляции, 54
XFree86, 54
добавление пользователей, 49
мышь, 51
пароль root, 47
программное обеспечение, 54
часовой пояс, 50
перезапуск, 55
разбиение жесткого диска, 39
с компакт-диска, 286
с магнитной ленты, 47
требования к аппаратным средствам, 30
устройства
EISA, 113
USB, 115
свопинга, 588
участки, 438

Ф

файловые системы и диски, 437
Soft Updates, 449
восстановление файловой системы, 84
грязные диски, 446
добавление новых жестких дисков, 460
кэширование каталогов в виртуальной памяти, 451
монтирование неродных файловых систем, 451
основы организации диска, 441
создание дискеты, 456
съемные носители и /etc/fstab, 454
файлы устройств, 437
файлы, снятие блокировок, 89
файлы устройств, 73, 439
физический уровень, 126
фильтрация пакетов, 219
базовые концепции, 210
ключевые слова и конфигурации, 213
принимать по умолчанию или отвергать, 209
протоколирование IPFilter, 218
разрешение доступа к сервисам, 213
реализация IPFilter, 219
флаги файлов, 186
флаги, поле, 514
функции учета, 248
функции, указываемые в каталоге Directory, 418

Х

хабы, 132

Ч

часовой пояс, 50
частичные обновления, 160
черные списки, отправители макулатурной почты, 387

Ш

шестнадцатеричные сетевые маски, 136
широковещание, Ethernet, 131
широковещательный адрес, 136
шифрование открытым ключом, 369
Certificate Authority, 368
запрос, создание, 365
сертификаты, 364
шифрование с одним ключом, 363

Э

эмуляция, 316

Я

ядро, 96–123
SMP и, 324
добавления в ядро, 119
загрузка с запасного ядра, 118
конфигурирование, 97–104
loader.conf, с помощью, 101
sysctl, 97–101
загрузчика вручную, 103

ядро
настройка производительности ядра, 121
настройка ядра, 105–115
более одного процессора, 113
записи устройств, 113, 164
основные параметры, 110
подготовка, 106
редактирование файлов ядра, 108
обновление ядра, 161
отладка ядра расширенная, 545
плохое ядро, 543
поддержка CIFS, 565
распространение, 123
создание ядра, 115
выявление неполадок, 117
ядро восстановления, 107

По договору между издательством «Символ-Плюс» и Интернет-магазином «Books.Ru-Книги России» единственный легальный способ получения данного файла с книгой «FreeBSD. Подробное руководство» (ISBN 5-93286-066-9) – покупка в Интернет-магазине «Books.Ru-Книги России». Если Вы получили данный файл каким-либо другим образом, Вы нарушили международное законодательство и законодательство Российской Федерации об охране авторского права. Вам необходимо удалить данный файл, а также сообщить издательству «Символ-Плюс» (www.symbol.ru), где именно Вы получили данный файл.