Денис Колисниченко

FreeBSD
ОТ НОВИЧКА К ПРОФЕССИОНАЛУ

Санкт-Петербург
«БХВ-Петербург»
2011

УДК 681.3.06
ББК 32.973.26-018.1
К60

Колисниченко Д. Н.

К60 FreeBSD. От новичка к профессионалу. — СПб.: БХВ-Петербург, 2011. — 544 с.: ил. — (В подлиннике)

ISBN 978-5-9775-0673-1

Материал ориентирован на последние версии операционных систем FreeBSD, PYC-BSD, OpenBSD. С позиции типичного пользователя BSD показано, как самостоятельно настроить и оптимизировать эту операционную систему.

Особое внимание уделяется повседневным задачам администратора. Рассмотрено резервное копирование, настройка сетевых сервисов, организация RAID-массивов, борьба с рекламными баннерами и спамом, анализ журналов сервера, подсчет трафика, мониторинг сети и др. Подробно описана настройка почтового сервера (Postfix), Web-сервера (Apache) в связке с интерпретатором PHP и сервером баз данных MySQL, серверов DNS/DHCP/FTP и других сетевых служб. Приведены рекомендации по защите рассмотренных в книге сетевых служб.

Для широкого круга пользователей FreeBSD

УДК 681.3.06
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.01.11.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 43,86.
Тираж 1500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12.

ISBN 978-5-9775-0673-1

© Колисниченко Д. Н., 2011
© Оформление, издательство "БХВ-Петербург", 2011

Оглавление

Введение ... **1**

ЧАСТЬ I. ЗНАКОМСТВО С СИСТЕМОЙ ... **3**

Глава 1. История UNIX и FreeBSD ... **5**

1.1. Первые операционные системы ... 5
1.2. Первые версии UNIX и их развитие ... 7
1.3. Появление BSD ... 8
1.4. Развитие UNIX-подобных систем ... 8
1.5. FreeBSD, OpenBSD и NetBSD ... 10
1.6. Имеет ли значение версия? ... 11

Глава 2. Установка FreeBSD ... **13**

2.1. Загрузка образов FreeBSD ... 13
2.2. Системные требования ... 14
2.3. Приступаем к установке ... 15
2.3.1. Загрузка с диска ... 15
2.3.2. Знакомство с программой установки ... 15
2.3.3. Разметка диска ... 19
2.3.4. Выбор загрузчика ... 21
2.3.5. Создание BSD-разделов внутри BSD-слайса ... 22
2.3.6. Установка программного обеспечения. Выбор источника установки ... 26
2.4. Постинсталляционная настройка системы ... 29
2.5. После перезагрузки ... 38

Глава 3. Установка OpenBSD ... **39**

3.1. Перед началом установки ... 39
3.2. Установка системы ... 40
3.2.1. Загрузка с компакт-диска ... 40
3.2.2. Настройка сети ... 41
3.2.3. Ввод пароля root ... 42

3.2.4. Перед разметкой жесткого диска ... 43
3.2.5. Разметка жесткого диска ... 44
3.2.6. Дистрибутивные наборы ... 47
3.2.7. Выбор часового пояса ... 48
3.2.8. Перезагрузка системы ... 49

Глава 4. Операционная система РУС-BSD: обзор, установка ... 51

4.1. Кратко о системе ... 51
4.2. Установка РУС-BSD ... 52

Глава 5. Резервное копирование. Быстрая переустановка BSD/Linux/Windows ... 55

5.1. Зачем нужно делать резервные копии? ... 55
5.2. Выбор носителя для резервной копии ... 55
5.3. Правила хранения DVD с резервными копиями ... 57
5.4. Стратегии создания резервной копии ... 58
5.5. "Живая" резервная копия ... 60
5.6. Клонирование дисков — программа Clonezilla ... 61
5.7. Программа tar ... 70
5.8. Сетевое резервное копирование ... 71
5.9. Запись DVD-болванок в FreeBSD ... 72

Глава 6. Создание LiveCD своими руками ... 73

6.1. Создание дампа. Команда *dump* ... 73
6.2. Создание LiveCD. Утилита FreeSBIE ... 74
6.3. Восстановление системы. Команда *restore* ... 75

Глава 7. Особые варианты установки FreeBSD ... 76

7.1. Использование LiveUSB. Установка FreeBSD на нетбук ... 76
7.2. Обновление до FreeBSD 8 ... 77
7.3. Установка по сети ... 78

ЧАСТЬ II. НАСТОЛЬНОЕ ПРИМЕНЕНИЕ BSD ... 81

Глава 8. Настройка консоли ... 83

8.1. Вход в систему ... 83
8.2. Понятие о работе в консоли ... 84
8.3. Виртуальные консоли ... 85
8.4. Правильное завершение работы в системе ... 86

8.5. Конфигуратор *sysinstall* ... 87
8.6. Файловый менеджер Midnight Commander ... 87
8.7. Изменение редактора по умолчанию ... 88
8.8. Использование редактора vi ... 89
8.9. Русификация консоли ... 92
8.10. Переход на UTF-8 ... 96

Глава 9. Установка графической среды GNOME ... 98

9.1. Графический интерфейс в FreeBSD ... 98
9.2. Установка портов и пакетов ... 99
9.3. Настройка запуска GNOME ... 101
9.4. Несколько слов о русификации системы ... 106
9.4.1. Добавление русской раскладки ... 106
9.4.2. О соответствии кодировок GNOME и консоли ... 109

Глава 10. Тонкая настройка графической подсистемы ... 111

10.1. Трюки с HAL ... 111
10.1.1. Включение <Ctrl>+<Alt>+<Backspace> ... 111
10.1.2. Запрет опроса устройств ... 112
10.1.3. Монтирование устройств с помощью HAL ... 113
10.2. Редактор конфигурации gconf-editor ... 114
10.3. Поддержка видеокарт NVIDIA ... 118

ЧАСТЬ III. КОМАНДНАЯ СТРОКА ... 119

Глава 11. Выбор и использование командной оболочки ... 121

11.1. Файл /etc/shells ... 121
11.2. Разнообразие выбора ... 122
11.2.1. Оболочка sh ... 122
11.2.2. Оболочка csh ... 122
11.2.3. Оболочка ksh ... 123
11.2.4. Оболочка bash ... 123
11.2.5. Оболочка zsh ... 124
11.2.6. Оболочка tcsh ... 125
11.2.7. Оболочка ash ... 125
11.2.8. Выбор оболочки ... 125
11.3. Оболочка bash ... 125
11.4. Оболочка tcsh ... 128
11.5. Перенаправление ввода/вывода ... 132

Глава 12. Создание сценариев на языке оболочки ... 134

12.1. Сценарии оболочки bash ... 134
12.1.1. Привет, мир! ... 134
12.1.2. Использование переменных в собственных сценариях ... 135
12.1.3. Передача параметров сценарию ... 136
12.1.4. Массивы и bash ... 137
12.1.5. Циклы ... 137
12.1.6. Условные операторы ... 138
12.1.7. Функции ... 139
12.1.8. Примеры сценариев ... 140
12.2. Сценарии оболочки tcsh ... 142
12.2.1. Переменные, массивы и выражения ... 142
12.2.2. Чтение ввода пользователя ... 144
12.2.3. Переменные оболочки, модификаторы форматов ... 144
12.2.4. Управляющие структуры ... 147

Глава 13. 24 полезные команды ... 152

13.1. Команда *man*: справочная система ... 152
13.2. Команда *uname*: информация о системе ... 152
13.3. Команда *clear*: очистка экрана ... 154
13.4. Команда *date*: вывод и установка даты и времени ... 154
13.5. Команда *exit*: выход из оболочки ... 154
13.6. Команда *passwd*: изменение пароля ... 154
13.7. Команда *uptime*: информация о работе системы ... 154
13.8. Команда *users*: информация о пользователях ... 155
13.9. Команды *w*, *who* и *whoami*: подробная информация о пользователях ... 155
13.10. Команда *diff*: сравнение файлов ... 156
13.11. Команда *grep*: текстовый фильтр ... 156
13.12. Команды *more* и *less*: постраничный вывод ... 156
13.13. Команды *head* и *tail*: вывод начала и "хвоста" файла ... 157
13.14. Команда *wc*: подсчет слов, строк и символов в файле ... 157
13.15. Команда *ftp*: стандартный FTP-клиент ... 157
13.16. Команды *links* и *lynx*: текстовые браузеры ... 159
13.17. Команда *md5*: вычисление контрольного кода MD5 ... 160
13.18. Команда *df*: информация об использовании дискового пространства ... 160
13.19. Команда *free* своими руками ... 161

ЧАСТЬ IV. АДМИНИСТРИРОВАНИЕ СИСТЕМЫ 163

Глава 14. Файловая система 165

14.1. Файловые системы, поддерживаемые FreeBSD 165
14.1.1. Производительность файловых систем 165
14.1.2. Какую файловую систему выбрать? 167
14.1.3. Интересные факты о ZFS 167
14.1.4. Монтирование UFS2 в асинхронном режиме 169
14.1.5. Включение SoftUpdates 169
14.2. Особенности файловой системы FreeBSD 170
14.2.1. Обо всем сразу: слайсы, разделы, блоки, иноды 170
14.2.2. Имена файлов в FreeBSD 173
14.2.3. Файлы и устройства 174
14.2.4. Корневая файловая система и монтирование 175
14.2.5. Стандартные каталоги FreeBSD 176
14.3. Команды для работы с файлами и каталогами 177
14.3.1. Работа с файлами 177
14.3.2. Работа с каталогами 179
14.4. Команда *ln*: создание ссылок 181
14.5. Команды *chmod*, *chown* и *chattr* 182
14.5.1. Команда *chmod*: права доступа к файлам и каталогам 182
14.5.2. Команда *chown*: смена владельца файла 184
14.5.3. Специальные права доступа (SUID и SGID) 184
14.5.4. Команда *chattr*: атрибуты файла, запрет изменения файла 185
14.6. Монтирование файловых систем 185
14.6.1. Команды *mount* и *umount* 185
14.6.2. Файлы устройств и их монтирование 186
14.6.3. Монтирование разделов при загрузке 191
14.7. Полезные примеры 192
14.7.1. Монтирование ISO-образа 192
14.7.2. Монтирование каталога 193
14.7.3. Проблемы со SCSI-приводами DVD 193
14.8. Добавление еще одного жесткого диска 193
14.9. Еще раз о Midnight Commander 200

Глава 15. Пользователи и группы. Квотирование 201

15.1. Многопользовательская система 201
15.2. Пользователь root 202
15.2.1. Максимальные полномочия 202
15.3. Создание, удаление и модификация пользователей 204
15.3.1. Создание пользователя: команда *adduser* 204

15.3.2. Удаление пользователя: команда *rmuser* 207
15.3.3. Изменение пароля пользователя: команда *passwd* 207
15.4. Подробно о создании пользователей 208
15.5. Группы пользователей 209
15.6. Ограничение дискового пространства 210

Глава 16. Загрузка и инициализация системы 213

16.1. Процесс загрузки FreeBSD 213
16.2. Сценарии инициализации 216
16.3. Планировщики заданий 217
16.4. Настройка синхронизации времени 219
16.5. Тюнинг системы с помощью файла sysctl.conf 219

Глава 17. Процессы 220

17.1. Аварийное завершение процесса 220
17.2. Программа top: кто больше всех расходует процессорное время? 223
17.3. Изменение приоритета процесса 226
17.4. Фоновое выполнение процессов 226

Глава 18. Установка программного обеспечения: порты и пакеты 227

18.1. Введение в пакеты и порты 227
18.2. Установка из портов 228
18.2.1. Установка порта 229
18.2.2. Удаление и переустановка порта 229
18.2.3. Установка коллекции портов 229
18.2.4. Обновление коллекции портов 231
18.2.5. Описание каталога /usr/ports 232
18.2.6. Обновление портов. Программа portupgrade 236
18.3. Установка программ из пакетов 238

Глава 19. Настройка печати 241

19.1. Системы печати lpr и CUPS 241
19.2. Принтеры и GDI-принтеры 241
19.3. Файлы описания принтеров 242
19.4. Установка CUPS 244
19.5. Установка принтера 244
19.6. Конфигурационные файлы CUPS 253

Глава 20. RAID-массивы ... 257

20.1. Что такое RAID? ... 257
20.2. Программные RAID-массивы ... 259
20.2.1. Программный RAID-массив на основе CDD ... 259
20.2.2. Программный RAID-массив на основе GEOM ... 261

Глава 21. Компиляция ядра ... 264

21.1. Установка исходных кодов ядра ... 264
21.2. Настройка ядра ... 265
21.2.1. Архитектура процессора ... 265
21.2.2. Создание копии файла конфигурации ядра ... 265
21.2.3. Редактирование файла конфигурации ядра ... 266
21.2.4. Включение PAE — поддержки более 4 Гбайт оперативной памяти ... 268
21.3. Сборка ядра ... 270

ЧАСТЬ V. СЕРВЕРНОЕ ПРИМЕНЕНИЕ BSD ... 271

Глава 22. Основы сетевого взаимодействия ... 273

22.1. Краткая история сетей ... 273
22.1.1. 1941–1975 годы ... 273
22.1.2. 1976–1982 годы ... 274
22.1.3. 1983–1989 годы ... 275
22.1.4. 1990–1995 годы ... 276
22.1.5. 1996–1999 годы ... 277
22.1.6. 2000 — наше время ... 278
22.2. Классификация сетей ... 278
22.2.1. По занимаемой территории ... 278
22.2.2. По топологии ... 279
22.2.3. По ведомственной принадлежности ... 281
22.2.4. По скорости передачи данных ... 281
22.2.5. По типу среды передачи данных ... 281
22.2.6. По способу организации взаимодействия компьютеров ... 282
22.3. Способы передачи данных в сетях ... 282
22.4. Модель OSI ... 283
22.5. Что такое протокол? ... 285
22.6. Адресация компьютеров ... 286
22.7. Система DNS ... 290
22.8. Монтаж Ethernet-сети ... 290
22.8.1. Развитие стандарта Ethernet ... 290

22.8.2. Несколько слов о коллизиях… 295
22.8.3. Монтаж сети 296
22.8.4. Ограничения при построении сети 301

Глава 23. Настройка локальной сети 304

23.1. Определение имени сетевого интерфейса 304
23.2. Настройка сетевого адаптера по DHCP 306
23.3. Настройка сетевого адаптера вручную 307
23.4. Настройка сетевого адаптера с помощью конфигуратора *sysinstall* 307
23.5. Настройка сетевого интерфейса с помощью команды *ifconfig* 309
23.6. Команда *route*: маршрутизация 311
23.7. Имя узла, IP-адреса серверов DNS 312
23.8. Несколько слов о поддержке IPv6 313
23.9. Суперсервер inetd 313
23.10. Команды диагностики сети 315

Глава 24. Настройка DSL-соединения 318

24.1. Причина популярности DSL-соединений 318
24.2. Физическое подключение ADSL-модема 318
24.3. Настройка соединения в FreeBSD 319
24.4. Управление переподключением 321

Глава 25. Подключение к сети Windows 322

25.1. Установка Samba 322
25.2. Файл конфигурации Samba 323
25.3. Настройка общих ресурсов 324
25.4. Оптимизация Samba 326
25.5. Программа smbclient 327

Глава 26. DHCP-сервер 328

26.1. Протокол динамической конфигурации узла 328
26.2. Конфигурационный файл DHCP-сервера 329
26.3. База данных аренды 331
26.4. Полный листинг конфигурационного файла 331
26.5. Привязка к MAC-адресу 332
26.6. Управление сервером DHCP 335
26.7. Настройка клиентов 335

Глава 27. DNS-сервер 336

27.1. Еще раз о том, что такое DNS 336
27.2. Запуск DNS-сервера 337

27.3. Файл конфигурации named.conf ... 338
27.4. Кэширующий сервер DNS ... 339
27.5. Полноценный DNS-сервер ... 343
27.6. Вторичный DNS-сервер ... 346
27.7. Обновление базы данных корневых серверов ... 347

Глава 28. Брандмауэр и шлюз ... 348

28.1. Что такое брандмауэр? ... 348
28.2. Перекомпиляция ядра ... 349
28.3. Конфигурация сети ... 350
28.4. Редактирование файла /etc/rc.conf ... 351
28.5. Редактирование файла /etc/rc.firewall ... 353
28.6. Создание отдельного файла правил ... 355

Глава 29. Прокси-сервер ... 356

29.1. Зачем нужен прокси-сервер в локальной сети? ... 356
29.2. Базовая настройка Squid ... 357
29.3. Практические примеры ... 359
29.3.1. Управление доступом ... 359
29.3.2. Создание черного списка URL ... 359
29.3.3. Отказ от баннеров ... 360
29.4. Управление прокси-сервером ... 360
29.5. Настройка клиентов ... 360
29.6. Отказ от баннеров с помощью редиректора Rejik ... 361
29.7. Анализатор протоколов Squid ... 363
29.8. Прозрачные прокси-серверы ... 364
29.8.1. Установка прокси-сервера OOPS ... 364
29.8.2. Прозрачный Squid ... 365
29.8.3. Проблемы с прозрачным Squid ... 366

Глава 30. FTP-сервер ... 369

30.1. Зачем нужен FTP? ... 369
30.2. Настройка стандартного ftpd ... 369
30.2.1. Запуск ftpd и проверка работоспособности ... 369
30.2.2. Настройка сервера ... 371
30.3. Сервер ProFTPD ... 373
30.3.1. Установка и запуск сервера ... 373
30.3.2. Конфигурационный файл сервера ... 373
30.3.3. Настройка реального сервера ... 376
30.4. Сервер vsftpd ... 378
30.4.1. Почему именно vsftpd? ... 378

30.4.2. Установка сервера vsftpd и всего необходимого ... 378
30.4.3. Создание базы данных MySQL ... 379
30.4.4. Конфигурационный файл сервера vsftpd ... 380

Глава 31. NFS — сетевая файловая система ... 382

31.1. Принцип работы NFS ... 382
31.2. Настройка и использование NFS ... 383
31.3. Монтирование экспортированной файловой системы на клиенте ... 385

Глава 32. Почтовый сервер ... 386

32.1. Выбор программного обеспечения ... 386
32.2. Установка вспомогательного ПО ... 387
32.2.1. Установка MySQL-сервера ... 388
32.2.2. Установка библиотеки Cyrus-sasl2 ... 389
32.2.3. Установка библиотеки Courier-authlib ... 389
32.2.4. Редактирование конфигурационных файлов ... 390
32.3. Установка Courier-IMAP ... 391
32.4. Установка postfix ... 393
32.5. Установка PostfixAdmin ... 399

Глава 33. Удаленный доступ по протоколу SSH ... 401

33.1. Протокол SSH и SSH-клиент ... 401
33.2. SSH-сервер ... 402

Глава 34. Web-сервер. Связка Apache + PHP + MySQL ... 406

34.1. Самый популярный Web-сервер ... 406
34.2. Установка Web-сервера, интерпретатора PHP, сервера MySQL ... 406
34.2.1. Установка Web-сервера Apache ... 406
34.2.2. Установка PHP ... 411
34.2.3. Установка MySQL-сервера ... 412
34.3. Управление серверами Apache и MySQL ... 412
34.4. Проблемы с запуском Apache ... 413
34.5. Тестирование настроек ... 414
34.6. Файлы конфигурации Web-сервера ... 416
34.6.1. Базовая настройка ... 416
34.6.2. Самые полезные директивы файла конфигурации ... 417
34.6.3. Директивы *Directory*, *Limit*, *Location*, *Files* ... 419
34.7. Оптимизация Apache ... 422
34.8. Пользовательские каталоги ... 424

Глава 35. Виртуальные частные сети ... 426

35.1. Для чего нужна виртуальная частная сеть? ... 426
35.2. Необходимое программное обеспечение ... 427
35.3. Соединение сеть-сеть ... 427
35.3.1. Постановка задачи ... 427
35.3.2. Выбор канала передачи данных ... 428
35.3.3. Перекомпиляция ядра ... 428
35.3.4. Установка ipsec-tools ... 429
35.3.5. Генерирование сертификатов ... 429
35.3.6. Редактирование файлов конфигурации ... 431
35.4. Соединение клиент-сеть ... 434
35.4.1. Выбор канала передачи данных ... 434
35.4.2. Перекомпиляция ядра ... 434
35.4.3. Установка порта poptop ... 435
35.4.4. Редактирование конфигурационных файлов ... 435
35.5. Настройка PPTP-клиентов ... 436
35.5.1. Настройка Linux-клиента ... 436
35.5.2. Настройка Windows-клиента ... 438

Глава 36. Защита сетевых сервисов ... 443

36.1. Защита Web-сервера ... 443
36.2. Защита FTP ... 444
36.3. Защита DNS ... 444
36.4. Защита Samba ... 446

ЧАСТЬ VI. ИНСТРУМЕНТЫ СИСТЕМНОГО АДМИНИСТРАТОРА ... 447

Глава 37. Системы мониторинга трафика ... 449

37.1. Простейшая система мониторинга трафика: darkstat ... 449
37.2. Система NeTAMS ... 452

Глава 38. Nagios — система мониторинга сети ... 456

38.1. Необходимость мониторинга сети ... 456
38.2. Установка Nagios ... 456
38.3. Настройка Nagios ... 458

Глава 39. Сниффер AimSniff — перехват ICQ-трафика пользователей ... 463

39.1. Юридические аспекты ... 463
39.2. Установка и настройка сниффера ... 464

Глава 40. Сканер nmap — программа аудита сети 466

40.1. Что такое nmap? 466
40.2. Установка nmap 467
40.3. Примеры использования nmap 467

Глава 41. Антивирусная проверка трафика 470

41.1. Постановка задачи 470
41.2. Установка HAVP и ClamAV 471
41.3. Настройка ClamAV и HAVP 472
41.4. Настройка Squid 473

Глава 42. SMS-рассылка 475

42.1. Постановка задачи 475
42.2. Установка SMS Tools 475
42.3. Русификация SMS 477

Глава 43. Шифрование разделов 479

43.1. Необходимость в шифровании 479
43.2. Технология gdbe 479
43.2.1. Включение gdbe 479
43.2.2. Шифрование нового жесткого диска 480
43.2.3. Монтирование уже зашифрованного жесткого диска 481
43.2.4. Автоматическое монтирование gdbe-устройств 482
43.3. Криптографическая файловая система geli 482
43.3.1. Особенности geli 482
43.3.2. Включение поддержки geli 482
43.3.3. Шифрование с помощью geli 483
43.3.4. Автоматическое подключение geli-устройств 483

ЧАСТЬ VII. ТЕОРИЯ И ПРАКТИКА СИСТЕМНОГО АДМИНИСТРИРОВАНИЯ 485

Глава 44. Стратегия администрирования 487

44.1. Структура IT-службы 487
44.2. И руководство, и пользователи довольны. Миф или реальность? 488
44.3. Роль главного администратора 491

Глава 45. Уход за "железом" 494

45.1. Обязанности администратора 494
45.2. "Про запас", или обменный фонд 495

45.3. Чистка компьютеров. Профилактика системы охлаждения ... 496
45.4. Охлаждение компьютеров ... 497
45.5. Стойки для оборудования ... 498
45.6. Влажность ... 499
45.7. Инструмент системного администратора ... 500

Вместо заключения ... 501

ПРИЛОЖЕНИЯ ... 503

Приложение 1. Двойная загрузка: Windows 7 и FreeBSD ... 505

Приложение 2. Настройка загрузчика GRUB: Linux и FreeBSD ... 507

Приложение 3. Проблемы с USB-накопителями в FreeBSD 8.0 ... 508

Приложение 4. Основные сетевые устройства ... 509

П4.1. Активное и пассивное сетевое оборудование ... 509
П4.2. Оборудование, необходимое для построения Ethernet-сети ... 509
П4.3. Оборудование, необходимое для построения сети Wi-Fi ... 513
П4.4. Дополнительные сетевые устройства ... 515

Предметный указатель ... 517

Введение

При подготовке этой книги мною преследовалась главная цель — не повторять официальное руководство по FreeBSD, с которым может ознакомиться любой желающий по адресу: **http://www.freebsd.org.ua/doc/ru_RU.KOI8-R/books/handbook/index.html**.

Данная книга — практическое руководство по настройке сервера (и, местами, — рабочей станции) на базе FreeBSD. Не скрою, что основной упор был сделан именно на серверное применение FreeBSD. И это не удивительно, ведь в большинстве случаев эта операционная система работает на серверах, а на домашних компьютерах и ноутбуках она встречается лишь при условии, что владелец такого домашнего компьютера или ноутбука — администратор сервера на базе FreeBSD или просто фанат этой системы.

Что же касается официального руководства, то наличие этой книги не освобождает вас от знакомства с ним — книга и официальное руководство будут отлично дополнять друг друга. Экспериментально установлено, что в руководстве допущены некоторые неточности, и в книге они будут разъяснены. С другой стороны, ряд моментов вообще в книге не рассматривается, поскольку при наличии официального руководства нет необходимости его дублировать.

Приведу небольшой пример. При настройке рабочей станции (именно рабочей станции, а не сервера) вам придется сконфигурировать графический интерфейс. В руководстве рассматривается настройка графической системы X Window System (которая уже давно называется X.Org — вот вам и первое отклонение руководства от истины) путем настройки конфигурационного файла xorg.conf. Но последние версии сервера X могут прекрасно работать вообще без конфигурационного файла — все настройки будут получены от демона HAL, создание и редактирование конфигурационного файла xorg.conf понадобится только в нештатных ситуациях. В книге формат файла org.conf не описан (в 90% случаев можно обойтись без него), зато он описан в руководстве. Если ваш случай из этих самых 10%, вы сможете из руководства узнать, как создать файл конфигурации сервера X. Зато в руководстве практически ничего не сказано о демоне HAL, а в книге приводится ряд примеров по настройке X средствами HAL.

Также в руководстве ничего не сказано о монтировании псевдофайловой системы procfs, которая необходима для нормальной работы графической среды GNOME — без нее пользователи не смогут войти в систему средствами GDM (графического

дисплейного менеджера GNOME). А что же делать, если вы настроили автоматический запуск GDM, перезагрузили компьютер, а после перезагрузки не можете попасть в систему? Что делать, если сервер X завис (такое иногда случается)? Ответы на эти вопросы вы найдете в книге — в руководстве почему-то они не рассматриваются. Не могу сказать, что это плохо, — ведь тогда я бы не написал книгу.

Излагаемый мной материал основан на последней (на момент написания книги) версии популярной операционной системы FreeBSD, версии 8.1. Однако практически все сказанное здесь будет актуально и для версии 7.*x*, и для будущей версии 9.0.

Кроме системы FreeBSD в книге уделяется внимание системам OpenBSD и PУC-BSD. Первая система считается самой защищенной в мире BSD-систем (хотя куда уже больше защиты?), а вторая обладает преднастроенным графическим интерфейсом пользователя, предустановленными пользовательскими приложениями и графическим инсталлятором, что позволяет использовать ее в качестве настольной BSD-системы. В остальном PУC-BSD подобна FreeBSD, поэтому все, что будет сказано о FreeBSD, актуально и для PУC-BSD.

Прежде чем вы приступите к чтению книги, хочется пояснить, как ее следует читать, и уточнить, какими знаниями должен обладать читатель. Ясно, что совсем начинающим пользователям эта книга будет мало чем полезна — они не смогут даже установить FreeBSD. Если вы считаете себя начинающим пользователем и установили FreeBSD, то можете "поднимать планку" — вы уже явно не новичок. Чтобы установить FreeBSD и работать с ней, вам нужно быть, как минимум, квалифицированным Windows-пользователем (чтобы вы знали хотя бы, что такое BIOS Setup и как записать ISO-образ на болванку и загрузиться с нее) или Windows-администратором. Знания Linux приветствуются — вам будет проще освоить FreeBSD. Хотя иногда проще научиться с нуля, чем переучиваться. Отмечу — наличие знаний Linux не является необходимым условием для работы с этой книгой. Даже если вы не имеете представления ни о файловой системе UNIX, ни о правах доступа, ни о сетевых сервисах — ничего страшного, обо всем этом вы из книги и узнаете.

Порядок чтения книги зависит от ваших знаний. Если вы уже работали с UNIX, можете начать читать с той главы, которая вам более всего интересна. В любой главе имеются ссылки на другие главы, в которых разъясняется тот или иной материал. Так что, если вам что-то непонятно, возможно, это уже было рассмотрено в предыдущих главах. Начинающим BSD-пользователям рекомендуется читать книгу последовательно. Впрочем, некоторые главы можно пропускать по своему усмотрению — например, главы по установке OpenBSD и PУC-BSD при отсутствии желания работать с этими операционными системами.

Вот теперь можно приступить к чтению книги.



ЧАСТЬ I

Знакомство с системой



Глава 1

История UNIX и FreeBSD

1.1. Первые операционные системы

Современный мобильный телефон — почти полноценный компьютер (если не считать неудобной клавиатуры и небольшого экрана). Но во времена появления первых операционных систем компьютеры весили тонны и занимали целые залы. И уж точно могу заверить, что первые компьютеры не были персональными. Наоборот, для обслуживания огромного компьютера и для проведения расчетов нанимался целый штат сотрудников.

На рис. 1.1 изображен компьютер Mark I, разработанный компанией IBM в 1944 году. В то время Mark I считался настоящим прорывом в будущее, началом эры "современных компьютеров".

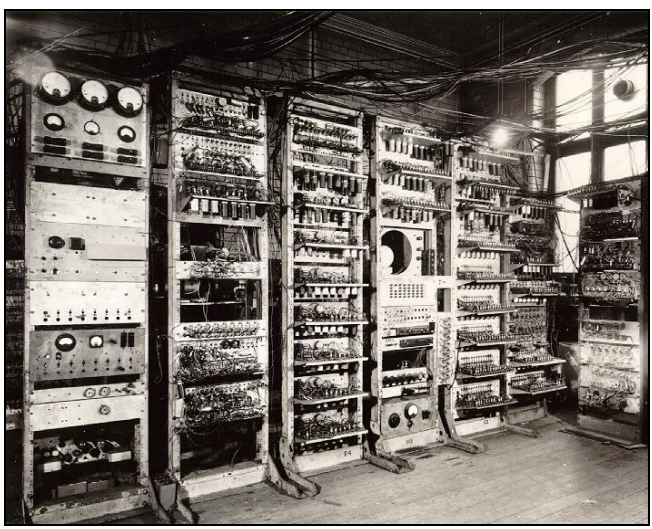


Рис. 1.1. Компьютер Mark I

Проблема компьютеров того времени заключалась в нерациональном использовании процессорного времени. Операционных систем не было как таковых, а на компьютере работали посредством ввода программы и данных с перфокарт.

Поэтому перед программистами стояла задача создать многозадачную операционную систему, которая бы могла обрабатывать несколько перфокарт сразу. Даже одновременная обработка двух перфокарт существенно бы сократила расчеты.

В конце 1950-х годов IBM выпускает свои новые компьютеры IBM 7090 и IBM 7094. Оба компьютера были предназначены для компании AT&T. Разработка операционной системы была поручена дочерней компании AT&T — Bell Labs. Процесс разработки начался в 1957 году под руководством Виктора Высотского (Victor A. Vyssotsky) — да, нашего соотечественника, родившегося в США в семье русского ученого, выехавшего из России в США в 1917 году. В результате для этих компьютеров была создана операционная система BESYS (Bell Operating System).

На рис. 1.2 изображен компьютер IBM 7090. Ввод данных в этот компьютер осуществлялся с перфокарт, а вывод — с помощью принтера (мониторов, как таковых, не было). Обратите внимание — операторы компьютера сидят не за мониторами, а именно за принтерами.



Рис. 1.2. IBM 7090

В 1964 году была начата разработка новой операционной системы Multics (MULTIplexed Information and Computing Service), поскольку старая операционная система BESYS уже не справлялась со своими задачами. Возглавил проект снова В. Высотский. Над проектом также работали Деннис Ритчи (Dennis MacAlistair Ritchie) и Кен Томпсон (Kenneth Thompson).

1.2. Первые версии UNIX и их развитие

Первая версия UNIX появилась в 1969 году, над ней работали три сотрудника компании Bell Labs: Кен Томпсон, Деннис Ритчи и Дуглас МакИлрой (Malcolm Douglas McIlroy). За основу была взята операционная система Multics. Новая операционная система называлась UNICS (UNIplexed Information and Computing System). Позже она была переименована в UNIX — американцы любят все упрощать.

Изначально UNIX создавалась для компьютера PDP-7 производства DEC. Этот компьютер также стал настоящим прорывом — ведь его производительность была на уровне гигантов от IBM, но он занимал гораздо меньше места — примерно как два холодильника (рис. 1.3), но не целую комнату.



Рис. 1.3. Компьютер DEC PDP-7

В ноябре 1971 года вышла версия UNIX V1 (UNIX Edition 1), именно с этой версии было принято решение нумеровать версии UNIX в формате V*n*, где *n* — номер редакции. Впрочем, самая первая версия UNIX, как ранее отмечалось, была разработана для компьютера PDP-7, а V1 разрабатывалась уже для компьютера PDP-11.

> ***Примечание***
>
> Некоторые пользователи ошибочно считают, что первая версия UNIX вышла 1 января 1970 года — именно с этой даты начинается отсчет времени в UNIX. Но на самом деле это не так, и дата 1 января 1970 года не имеет ничего общего с датой выхода какой-либо версии UNIX.

Чтобы адаптировать UNIX под компьютер PDP-11, разработчикам пришлось ее практически полностью переписать с нуля, поскольку версию UNIX для PDP-7 запустить на компьютере PDP-11 было невозможно. Ведь первая UNIX была написана на ассемблере, что жестко привязывало ее к конкретной архитектуре компьютера.

Параллельно работе над UNIX Кен Томпсон разрабатывал язык высокого уровня B. Хотя сам язык был создан в 1969 году, в состав UNIX он был включен в 1972 году, когда вышла UNIX V2 — вторая редакция UNIX. Версия языка B, разработанного в 1969 году, была интерпретирующей (как язык BASIC), что не лучшим способом сказывалось на производительности. Поэтому в 1973 году Кен Томпсон и Деннис Ритчи модифицировали язык B — теперь он стал компилирующим, что позволило существенно повысить производительность программ, написанных на этом языке. Новая версия языка была названа C. Компилятор языка C впервые вошел в состав UNIX в 1973 году — в версию V3.

Несмотря на наличие в операционной системе компилятора C, сама операционная система была по-прежнему написана на ассемблере. Но разработчики понимали, что ассемблер тормозит развитие системы, и приняли решение переписать систему на языке C. В октябре 1973 года выходит редакция UNIX V4 — уже частично переписанная на языке C.

В 1974 году появилась версия V5, не содержавшая ничего интересного, а вот в 1975 году вышла версия V6, знаменитая книгой Джона Лайонса (John Lions) "Commentary on UNIX 6th Edition, with Source Code" (Комментарии к 6-й редакции UNIX, с исходным кодом). В этой книге, изданной в 1976 году, подробно рассматривались принципы работы UNIX, а также был представлен исходный код ее шестой версии.

В 1979 году вышла редакция V7 — последняя единая версия UNIX от AT&T. Фактически, 1979 год можно считать началом великого раскола в разработке UNIX.

1.3. Появление BSD

Операционная система UNIX распространялась абсолютно бесплатно, поскольку компания AT&T не имела права заниматься коммерческой деятельностью. Одна из таких копий попала в университет Беркли. Там на базе AT&T UNIX V6 создали первую версию BSD (Berkeley Software Distribution) UNIX (или 1BSD). Это произошло в 1978 году, а к 1979 году было выпущено целых две версии BSD: 2BSD и 3BSD (последняя основана на UNIX V7).

Аспиранты университета Беркли разработали для BSD новую командную оболочку, текстовый реактор (похожий на vi), а также усовершенствовали механизм управления памятью.

В отличие от AT&T, университет Беркли имел право заниматься коммерческой деятельностью и продавал свою систему BSD по 50 долларов за одну копию. Так программное обеспечение оказалось разделенным на две группы: бесплатное (от AT&T) и коммерческое (от BSD).

1.4. Развитие UNIX-подобных систем

В 1980-е годы разработка операционных систем развивалась лавинообразно. И за создание собственных систем, в том числе и на базе UNIX (не важно: AT&T или BSD), взялись десятки компаний.

Примечание

С неполным древом развития UNIX-подобных систем вы можете ознакомиться по адресу: **http://upload.wikimedia.org/wikipedia/commons/d/d9/Unix_history-simple.en.svg**[1]. Это древо настолько большое, что опубликовать в книге его было бы проблематично, — проще посмотреть его в электронном виде. А почему оно неполное? Да потому, что Linux — это тоже в некотором роде UNIX, а представить на одном древе все возможные дистрибутивы Linux, число которых растет с каждым днем, довольно сложно.

Так, Microsoft "прославилась" своей крайне неудачной реализацией — Xenix, которая не имела коммерческого успеха и была продана (имеется в виду, что были проданы права на эту систему) компании SCO. Далее эта система стала известна как SCO UNIX[2], а в 1994 году ее переименовали в OpenServer. Система существует и по сей день — видимо, что не удалось Microsoft, получилось у SCO. И это не пустые слова — на SCO UNIX (имеется в виду не конкретная операционная система, а линейка UNIX-систем от компании SCO[3]) работают более 40% компьютеров рынка аптек США, 10 крупных международных компаний выбрали именно SCO UNIX, эта система задействована в более чем 12 000 "ресторанов" McDonald's. SCO UNIX используется также концерном BMW и установлена на компьютерах более чем 22 000 отделений Банка России. Очевидно, Microsoft, учитывая такой успех системы, теперь "кусает локти"…

Примечание

Почему слово "ресторан" — в кавычках? Вспоминаю фразу известного сатирика: "Самый короткий анекдот про Америку: McDonald's — это ресторан".

Компания HP также разработала свою UNIX — HP-UX. Ей в свое время занимался Стив Джобс — будущий основатель компании Apple. Кстати, вы будете удивлены, но в основе Mac OS — BSD…

В 1980-м году операционная система BSD UNIX была выбрана для своих компьютеров агентством по перспективным оборонным научно-исследовательским разработкам США (The Defense Advanced Research Projects Agency, DARPA). Основная причина выбора именно этой системы, а не UNIX V7 — наличие в ней усовершенствованной поддержки протокола TCP/IP. UNIX V7 также поддерживала TCP/IP, но поддержка этого протокола в BSD была реализована лучше.

В 1983 году компания AT&T получает право заниматься коммерческой деятельностью. В этом же году выходит новая версия UNIX, но теперь версии нумеруются иначе — UNIX System V R*n*, где *n* — номер версии. Все версии R*n* стали коммерческими — больше AT&T никому не сделает такого подарка, но, учитывая успех BSD, с этим решением она опоздала…

В 1989 году вышла версия System V R4, в которой наконец-то появилась нормальная поддержка TCP/IP, сетевая файловая система NFS, символические ссылки, новые командные интерпретаторы и много других полезных вещей.

---

[1] Файлы формата SVG можно открыть разными программами — например, программой Adobe SVG Viewer или браузером Chrome, который поддерживает формат SVG по умолчанию.

[2] Ознакомиться с системой SCO UNIX можно по адресу: **http://www.ru.sco.com/**.

[3] Кстати, кроме OpenServer есть еще SCO UnitWare, а все вместе это и есть SCO UNIX, см. **http://www.ru.sco.com/products/unix/**.

В 1990 году System V преобразовывается в SCO UnixWare (это еще одна UNIX-система от компании SCO). Эта версия дожила до наших дней, причем даже не изменила название[1].

На базе System V была построена и операционная система Solaris. Сначала ее поддерживала Sun Microsystems, а сейчас — Oracle Corporation.

1.5. FreeBSD, OpenBSD и NetBSD

Началом разработки FreeBSD (как OpenBSD и NetBSD) считается создание Биллом Джолитцем (Bill Jolitz) системы 386BSD — версии BSD, предназначенной для процессора Intel 80386. Со временем над системой 386BSD начали трудиться другие разработчики, между которыми в 1993–94 годах возникли разногласия. Одни программисты хотели создать простую операционную систему, доступную обычным пользователям. Основная их идея заключалась в том, чтобы использовать операционную систему мог любой пользователь , а не только программист-"гуру". По идее должна была получиться "пользовательская" версия BSD для компьютеров с архитектурой *x*86 — в то время процессор 80386 уже устаревал, а новая разработка (80486) набирала обороты, поэтому привязываться к 386 было нельзя. Вторая группа разработчиков хотела создать универсальную систему на базе BSD. Но универсальность ее должна была заключаться в поддержке любой аппаратной платформы — чтобы в мире не было платформы, на которой бы не могла запуститься их система.

По сути, обе команды хотели создать универсальную систему, вот только понятие об универсальности у каждой команды было разное. В результате произошел раскол команды 386BSD и появились две новые системы: FreeBSD и NetBSD. Первая была ориентирована только на архитектуру *x*86 и распространялась бесплатно. Вторая система тоже удалась. Сейчас ее текущая версия 5.0.2 поддерживает 57 разных аппаратных платформ[2]. Действительно, такой универсальностью может похвастаться не любая система. Но именно поэтому NetBSD никогда не будет бесплатной, поскольку не все производители "железа" готовы бесплатно предоставлять подробную информацию о своей архитектуре.

В 1995 году на базе NetBSD была создана бесплатная операционная система OpenBSD. Основная идея этой системы — открытость, что понятно из названия. Исходный код этой системы открыт и доступен всем желающим, он также не использует "закрытый" код других систем[3].

Вернемся к FreeBSD. Первая ее версия увидела свет в самом конце 1993 года, но уже в мае следующего года появилась версия 1.1. В настоящее время ведется разработка 9-й версии FreeBSD, но когда выйдет ее релиз, пока не известно. Может,

---

[1] Ознакомиться с системой SCO UnixWare можно по адресу: **http://www.ru.sco.com/products/unixware/714/**.

[2] Ознакомиться с системой NetBSD можно по адресу: **http://www.netbsd.org/**.

[3] Информация о системе OpenBSD доступна на сайте: **http://www.openbsd.org**.

это произойдет в конце 2010 года, а может — в 2011 году. Рассматривать особенности ранних версий не вижу смысла, поскольку они уже неактуальны. Тем более, что сами разработчики FreeBSD сейчас рекомендуют ориентироваться на версии 8.1 или 7.2/7.3.

1.6. Имеет ли значение версия?

Поскольку обычно FreeBSD используется на сервере (исключение могут составить лишь компьютеры фанатов этой операционной системы), особой разницы, какую версию вы будете использовать, — нет. Учитывайте только поддерживаемое той или иной версией FreeBSD "железо" и ее новые возможности — нужны ли они вам? Список совместимого с версией 8.1 "железа" можно просмотреть по адресу: **http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/hardware.html**.

Почему у меня такое отношение к версии операционной системы? Посудите сами. Возьмем обычный домашний компьютер, пусть даже ноутбук. В нем имеются два стандартных для любого современного ноутбука устройства: веб-камера и адаптер Wi-Fi. К примеру, веб-камера не заработала в Mandriva 2009. Что делать? Ждать выхода 2010-й версии (предположим, что ее пока нет) или переходить на другой дистрибутив. Хорошо, перейдем на Ubuntu 8. И что? В нем работает веб-камера, но отказывается работать адаптер Wi-Fi. А тут уже подоспела версия 2010 дистрибутива Mandriva, в которой и веб-камера, и беспроводной адаптер работают. Какую версию вы выберете? Конечно же, 2010. Короче, придется поработать головой и попробовать несколько разных дистрибутивов.

А вот с сервером все проще. На нем не будет веб-камеры и других сугубо пользовательских устройств. Процессор, оперативная память, жесткий диск или аппаратный RAID-массив жестких дисков — все! Поддержка видеокарты для сервера особо не нужна, как и звуковой платы, — в игры на нем играть никто не собирается. А вывести текст на экран монитора можно и без трехмерных возможностей видеокарты.

При выборе версии операционной системы для сервера важно совсем другое: чтобы ОС полностью поддерживала установленный процессор, объем оперативной памяти и т. д. К слову, если у вас многопроцессорная (SMP) система, то нужно выбрать версию FreeBSD не ниже 6.0, потому что именно в этой версии появились поддержка SMP и улучшенная поддержка беспроводных сетей — это на случай, если вы надумаете развернуть беспроводную сеть с сервером на базе FreeBSD. Понятно, что сейчас версия 6.0 выглядит очень устаревшей на фоне версий 8.1 и приближающейся 9.0.

Итак, если у вас современный сервер, выбирайте версию 8.1, но смотрите, чтобы сборка FreeBSD соответствовала архитектуре вашего процессора. А вот при наличии старенького компьютера, выбирать версию придется более тщательно. Ведь и слишком старую систему устанавливать не хочется, но и последняя версия как бы не нужна… Владельцам старых однопроцессорных машин я бы посоветовал вер-

сию 7.1. В ней оптимизирована поддержка протокола SCTP (Stream Control Transmission Protocol), файловая система UFS2 стала журналируемой (а значит, менее чувствительной к сбоям), включен компилятор gcc версии 4.2. А вот владельцам старых многопроцессорных машин лучше выбрать версию 7.2, в который в очередной раз улучшена поддержка SMP, появилась полная поддержка файловой системы ZFS (Zettabyte File System). Также версию 7.2 можно порекомендовать счастливым обладателям редких (но не очень новых) машин на базе процессоров UltraSPARC III ("Cheetah") и SPARC64. Эта версия полностью поддерживает такие процессоры, а также объем ОЗУ до 6 Гбайт (для 64-битных процессоров) и до 3,6 Гбайт (для 32-битных) У вас еще больше оперативной памяти? Тогда вам нужно собрать так называемое PAE-ядро, о чем мы поговорим в *главе 21*.

В восьмой версии в очередной раз улучшена поддержка SMP (ее улучшают с каждой версией — все-таки серверная операционная система), исправлены ошибки в реализации механизма работы с USB-устройствами, обеспечена полная поддержка ZFS (теперь можно даже загружаться с ZFS-раздела), появился монитор виртуальных машин Xen DomU, улучшена система "песочниц" (jails)[1].

Теперь, когда вы кратко познакомились с историей мира UNIX, можно перейти к установке FreeBSD, которая будет рассмотрена в *главе 2*.

[1] Кроме всего этого, в 8-й версии FreeBSD появилось много приятных нововведений, с которыми вы можете ознакомиться в статье "Чертенок из табакерки" по адресу: **http://www.dkws.org.ua/phpbb2/viewtopic.php?p=30335**.



Глава 2

Установка FreeBSD

2.1. Загрузка образов FreeBSD

Получить образы FreeBSD может любой желающий — это не тайна за семью замками. Скачайте их с FTP-сервера: **ftp://ftp.freebsd.org/pub/FreeBSD/releases**.

Затем выберите архитектуру (я для большей совместимости выбрал i386), перейдите в каталог **ISO-IMAGES**, а затем — в каталог, соответствующий номеру версии. Например, ISO-образы последней версии (8.1) FreeBSD для архитектуры i386 находятся по адресу: **ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.1/** (рис. 2.1).

Содержание /pub/FreeBSD/releases/i386/ISO-IMAGES/8.1/

Имя	Размер	Последнее изменение
[родительский каталог]		
CHECKSUM.MD5	386 B	19.07.10 5:19:00
CHECKSUM.SHA256	561 B	19.07.10 5:19:00
FreeBSD-8.1-RELEASE-i386-bootonly.iso	45.7 MB	19.07.10 3:26:00
FreeBSD-8.1-RELEASE-i386-disc1.iso	645 MB	19.07.10 3:27:00
FreeBSD-8.1-RELEASE-i386-dvd1.iso.gz	1.9 GB	19.07.10 3:28:00
FreeBSD-8.1-RELEASE-i386-livefs.iso	250 MB	19.07.10 3:28:00
FreeBSD-8.1-RELEASE-i386-memstick.img	904 MB	19.07.10 3:24:00

Рис. 2.1. Содержимое каталога ISO-IMAGES/8.1

Какой из образов выбрать? Я бы посоветовал не жалеть трафик и скачать образ FreeBSD-8.1-RELEASE-i386-dvd1.iso.gz. Понимаю, что 1,9 Гбайт — это не 645 Мбайт, но вы получите нормальный установочный диск с FreeBSD и сможете устанавливать с этого диска так же и программные пакеты. А если вы выберете образ размером 645 Мбайт, то систему установите, но программы все равно придется закачивать из Интернета.

Образ размером 645 Мбайт приходится выбирать, если планируется установка на старый компьютер, не оснащенный приводом DVD, поскольку такой образ можно записать на обычную CD-болванку.

Любителям всякого рода экспериментов можно посоветовать CURRENT-ветку FreeBSD 9, доступную по адресу: **ftp://ftp.freebsd.org/pub/FreeBSD/snapshots/201009/FreeBSD-9.0-CURRENT-201009-i386-dvd1.iso**. Но FreeBSD версии 9 нельзя использовать для организации реального сервера — она "сыровата". А вот для экспериментов на "лишней" или же виртуальной машине — всегда пожалуйста.

Модель разработки FreeBSD

Чем отличаются ветки CURRENT, STABLE и RELEASE? Текущая ветка, над которой работают в данный момент разработчики FreeBSD, называется CURRENT. Номер текущей версии сейчас — 9. В CURRENT помещаются все желаемые изменения. Когда разработчики решат, что вроде бы исправили все "баги", тогда они выпускают так называемую *стабильную* версию — STABLE. В STABLE помещается все то, что прошло проверку в CURRENT, если та или иная возможность работала нестабильно, в STABLE она не помещается. STABLE-ветку тестируют независимые пользователи, release-инженеры и сами разработчики. Затем она превращается в RELEASE-версию. Фактически, ветка RELEASE — это тщательно протестированная ветка STABLE.

Итак, скачайте и распакуйте образ FreeBSD. Он запакован архиватором gz, поэтому проблем с распаковкой быть не должно — вы распакуете такой архив и в Windows, и в Linux. Записать образ на диск можно любой программой для прожига дисков: хоть Nero, хоть встроенными средствами записи ISO-образов Windows 7. Подробные инструкции приводить не стану — если вы не знаете, как записать образ на диск, то срочно покупайте другую книгу, где описаны основы компьютерной грамотности. Да и про FreeBSD в этом случае лучше на некоторое время забыть. Можно попробовать UNIX-подобную ОС попроще, например, Linux Ubuntu.

2.2. Системные требования

О системных требованиях говорить особо нечего. Скорее всего, вы не найдете компьютер, на который нельзя было бы установить FreeBSD 8. Что же касается места на диске, то тут все зависит от выбранного для установки дистрибутива. Для минимальной установки FreeBSD достаточно 1 Гбайт дискового пространства (если окажется меньше, программа установки не позволит установить FreeBSD, хотя минимальная установка реально занимает менее 1 Гбайт), но проблема заключается в том, что на жесткий диск объемом 1 Гбайт вы установить систему сможете, но не сможете настроить сервер. Ведь для превращения вашего компьютера в сервер нужно будет установить также и соответствующие программы. Но давайте смотреть правде в лицо — вы не станете устанавливать FreeBSD на компьютер, на котором установлена Windows, следовательно, сможете использовать весь жесткий диск. А весь жесткий диск сегодня это как минимум 8 Гбайт (при установке на очень несовременный компьютер или нетбук), чего должно хватить.

С оперативной памятью ситуация такая же. Необходимый минимум составляет 32 Мбайт, но на любом нормальном (я не говорю — современном) компьютере сейчас установлено 256 Мбайт. Вы можете найти в углу запылившийся Pentium с 64 Мбайт оперативной памяти и жестким диском на 20 Гбайт — этого более чем достаточно для установки FreeBSD.

2.3. Приступаем к установке

2.3.1. Загрузка с диска

Войдите в BIOS Setup (обычно для этого используется клавиша <Del>, но иногда приходится нажать <F2>, <F10> или, например, комбинацию клавиш <Alt>+<S>). Включите загрузку с привода DVD — обычно для этого нужно изменить значение параметра **Boot Sequence** или **Boot Order**.

> ***Внимание!***
>
> При установке FreeBSD на компьютер с процессором AMD 64 и установленной видеоплатой от NVIDIA, отключите ACPI в BIOS или выберите при загрузке с установочного диска пункт **2. Boot FreeBSD with ACPI disabled** (см. также *разд. 2.3.2*, *16.1*).

Я рекомендую вам не спешить, приступая к установке. Принцип "Veni, vidi, vici" ("Пришел, увидел, победил") с FreeBSD не работает. Взять с наскока даже программу установки у вас не получится. Можно запросто "наломать дров", и систему придется переустанавливать, возможно, даже не один раз. А если вы устанавливаете FreeBSD на компьютер, где уже установлена другая операционная система (но зачем?!), следует быть вдвойне внимательным и перед установкой сделать резервную копию всех важных данных. О двойной загрузке мы поговорим в *приложениях 1* и *2*.

2.3.2. Знакомство с программой установки

После загрузки с инсталляционного диска, вы увидите меню загрузчика (рис. 2.2). Для начала установки просто нажмите клавишу <Enter>. Если необходимо отключить ACPI, нажмите клавишу <2>.



Рис. 2.2. Меню загрузчика FreeBSD



Рис. 2.3. Выбор страны

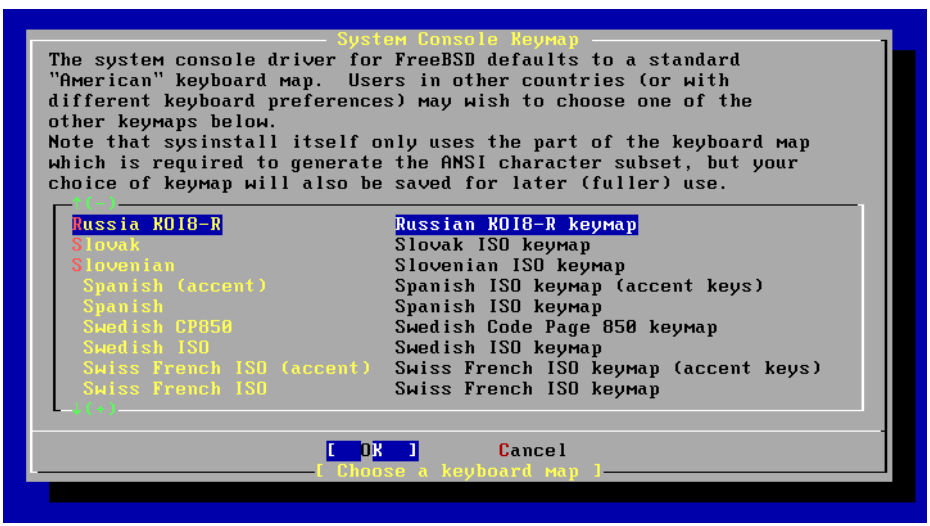


Рис. 2.4. Выбор раскладки клавиатуры

Первым делом программа установки предложит выбрать страну (рис. 2.3) и используемую раскладку (рис. 2.4).

Далее вы увидите основное меню программы установки (рис. 2.5), содержащее следующие команды:

- **Usage** — краткое руководство по использованию системы меню;
- **Standard** — стандартная установка. Рекомендуется, если вы устанавливаете FreeBSD в первый раз и желаете в процессе установки настроить сеть, добавить

учетные записи пользователей, установить пароль root и выполнить другие действия по базовой настройке системы;

- **Express** — экспресс-установка, подходит для экспертов. Не выбирайте этот вариант, если вы ни разу не устанавливали FreeBSD. В процессе установки задается гораздо меньше вопросов, что не может не радовать, но, с другой стороны, не настраивается сеть, не добавляются учетные записи пользователей. Поэтому вы должны знать, как выполнить потом все необходимые настройки самостоятельно;
- **Custom** — пользовательская установка, тоже прерогатива экспертов;
- **Configure** — постинсталляционная настройка FreeBSD. Как вы сами понимаете, эту команду нужно выбирать после установки FreeBSD (вызвать установщик можно командой `sysinstall`);
- **Doc** — различная документация по установке FreeBSD. Учитывая, что у вас есть эта книга, такая команда вам не понадобится;
- **Keymap** — позволяет выбрать раскладку клавиатуры, но поскольку мы это уже сделали, такая команда вам тоже не понадобится;

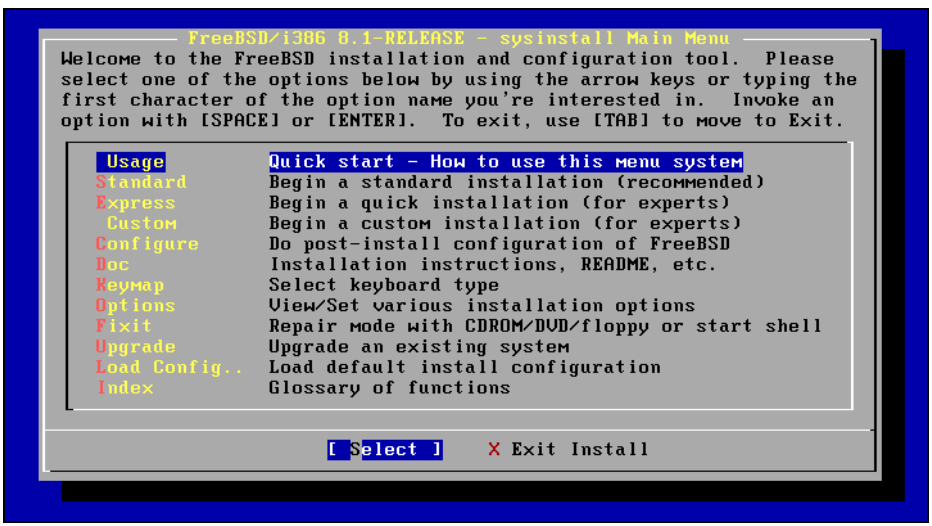


Рис. 2.5. Основное меню программы установки

- **Options** — позволяет просмотреть/установить различные опции программы установки (рис. 2.6). В частности, здесь вы можете выбрать носитель пакетов: либо **CDROM** — для установки с DVD, либо **FTP** — для установки с FTP-сервера **ftp.freebsd.org**. Сами понимаете, без доступа к Интернету установить пакеты с FTP-сервера не получится. Так что в этом случае придется предварительно настроить сеть. Хорошо, если в вашей сети есть DHCP-сервер — тогда просто установите значение **YES** для опции **DHCP**. В противном случае возвращайтесь в главное меню, выбирайте команду **Configure** и настраивайте сеть по полной. Подробно об установке по сети мы поговорим в *главе 7*;

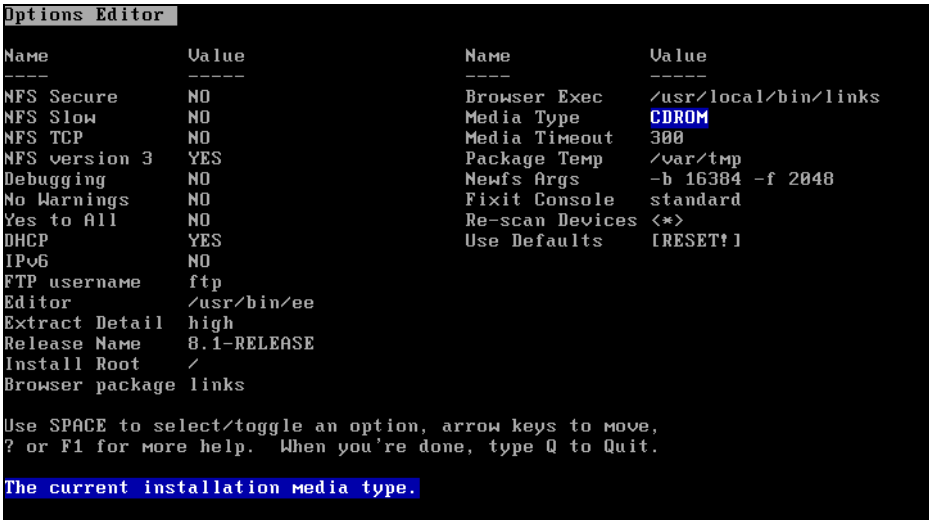


Рис. 2.6. Параметры установки

- **Fixit** — вы перейдете в режим восстановления или просто у вас появится возможность запустить оболочку. Используется для восстановления уже установленной системы, которая отказывается загружаться;
- **Upgrade** — обновление уже установленной системы;
- **Load Config…** — загружает конфигурацию установки;
- **Index** — глоссарий функций.

Итак, выберите **Standard** для продолжения установки. Система поставит вас перед фактом, что придется использовать не очень удобную программу fdisk, но другого выбора у нас нет, о чем свидетельствует кнопка **OK** (рис. 2.7).



Рис. 2.7. Будем использовать программу fdisk

Примечание

Бывшие Linux-пользователи будут, наверное, повергнуты в шок! Программа установки самой современной версии FreeBSD напоминает программу установки дистрибутива Red Hat Linux 6-й или даже 5-й версии (с более ранними я не работал). Но 6-я версия Red Hat — это 1999 год. Выходит, что FreeBSD по комфорту отстала от Linux более чем на 10 лет. Так и есть, но ничего с этим не поделаешь.

2.3.3. Разметка диска

Перед началом разметки диска нужно разобраться с организацией разделов жесткого диска применительно к FreeBSD. Все, что вы знали до этого, следует забыть. Нам было известно, что один физический диск можно разбить на логические диски, тома или разделы (в англ. терминологии — partitions). Когда нужно установить Linux, вы уменьшаете размер Windows-раздела и на освободившемся месте создаете Linux-раздел.

То, что мы называли в Windows и Linux *разделом* (partition), в FreeBSD называется *слайсом* (slice). При установке Linux надо было создать несколько разделов — как минимум два: один для корневой файловой системы, другой — для подкачки. В FreeBSD нужно создать всего лишь один слайс. Но сам слайс делится на разделы (partitions). В FreeBSD раздел — это часть слайса, внутреннее преставление слайса, а не раздел физического жесткого диска.

Если в вашей системе несколько жестких дисков, то перед запуском fdisk система позволит вам выбрать физический диск, на который будет произведена установка системы.

Теперь я должен предупредить вас об особенности FreeBSD, связанной с определением жестких дисков. Представим, что мы используем старые добрые IDE-диски. Впрочем, учитывая, что FreeBSD часто из соображений экономии средств устанавливается на старенький компьютер, то такая ситуация — не редкость. В системе может быть два контроллера: Primary и Secondary. К каждому контроллеру могут подсоединяться по два устройства, то есть всего можно подключить к компьютеру четыре устройства: Primary Master, Primary Slave, Secondary Master, Secondary Slave. Этим устройствам в FreeBSD соответствуют имена ad0, ad1, ad2 и ad3. Имена назначаются жестко, а не по мере определения устройства.

Примечание

SCSI-дискам в FreeBSD присваиваются имена da*N*, где *N* — номер диска.

Предположим, что у нас есть два диска, подключенные как Primary Master и Secondary Master. Им будут назначены имена ad0 и ad2. Если вы подключите третий диск как Primary Slave, ему будет назначено имя ad1. Некоторые операционные системы (например, Windows) определяют диски по мере их подключения. Если бы FreeBSD работала так же (к счастью, она работает иначе), то при наличии дисков Primary Master и Secondary Master им сначала были бы назначены имена по порядку: ad0 и ad1. Но когда бы вы подключили Primary Slave, ему бы было назначено имя ad1, а диску Secondary Master переназначено новое имя — ad2, что создало бы некую путаницу. Система искала бы данные, которые раньше были на ad1, но с пе-

ременой имени диска сейчас они оказались на ad2! Поэтому FreeBSD жестко привязывает имена дисков, что, как мы уже убедились, очень удобно.

Слайсам в FreeBSD назначаются имена ad*X*s*N*, где *X* — номер диска, *N* — номер слайса. Посмотрите на рис. 2.8 — там изображен диск, где уже имеется один BSD-слайс (описание **freebsd**) — ad0s1. Остальное пространство — не используется. Для Windows-слайса указывается описание **fat** или **ntfs**.

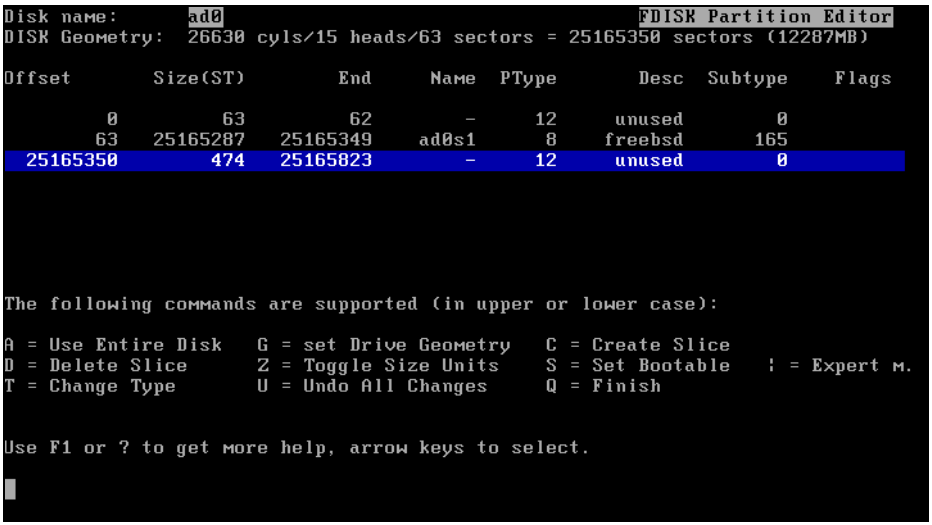


Рис. 2.8. Программа fdisk

Программа fdisk не умеет изменять размеры слайсов. Если у вас уже есть Windows-слайсы, и вы хотите на такой диск установить FreeBSD, тогда можно предложить следующие варианты:

- воспользоваться предварительно программой Partition Magic или подобной — с помощью такой программы разметки диска вы сможете уменьшить размер Windows-слайса без потери данных. Затем запустите установку FreeBSD, запустите программу fdisk, выберите неиспользуемое пространство (описание **unused**), нажмите клавишу <C> для создания нового слайса и введите его размер (рис. 2.9). Размер удобнее вводить с модификатором `M` (что означает мегабайты), например, `2048M` — это 2 Гбайт. Тип слайса можно изменить нажатием клавиши <T>. Но это нужно делать только, если вы собираетесь создать слайс для другой системы: Linux или Windows. По умолчанию fdisk создает BSD-слайсы (рис. 2.10);
- запустив программу fdisk, нажать клавишу <A> для использования всего диска — поскольку мы будем настраивать сервер (а зачем FreeBSD на клиентской машине?), то это оптимальный вариант. К чему ломать голову над разметкой диска, если можно использовать сразу весь диск? Вот только при наличии на диске каких-то данных было бы хорошо на всякий случай сделать их копию — ведь после нажатия клавиши <Q> все данные будут удалены;

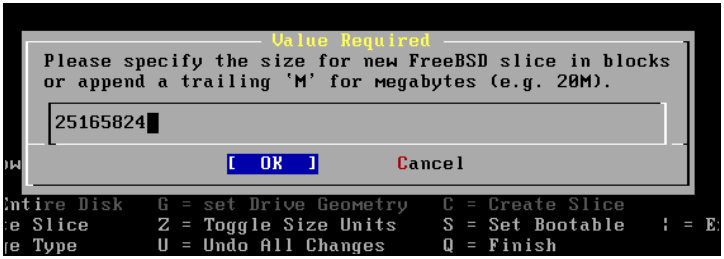


Рис. 2.9. Ввод размера нового слайса

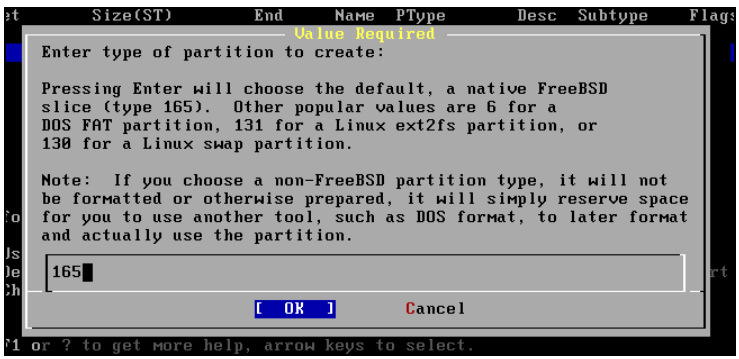


Рис. 2.10. Изменение типа слайса

- удалить один из слайсов — сделать это можно, нажав клавишу <D>. Используемое удаленным слайсом пространство будет помечено как неиспользуемое (**unused**), дальше на его месте можно создать новый слайс, нажав клавишу <C>, как было показано раньше.

Думаю, суть вы уловили. Даже если на диске имеются какие-то данные, я рекомендую перенести их на другой жесткий диск или на DVD, затем загрузиться с инсталляционного диска и нажать клавишу <A> в программе fdisk.

> ***Примечание***
>
> Кстати, на рис. 2.8 как раз и показан пример разметки виртуального жесткого диска размером 12 Гбайт. Диск создан в VMware — а как иначе сделать скриншоты программы установки, не будешь же фотографировать монитор?

Для продолжения изменений нажмите клавишу <Q>. Для отмены изменений — клавишу <U>.

2.3.4. Выбор загрузчика

Далее программа установки предложит вам выбрать загрузчик (рис. 2.11). Задача загрузчика — загрузка ядра операционной системы и передача ему управления. Вам доступны три варианта:

- **Standard** — неинтерактивный загрузчик. Его можно использовать, если на компьютере нет других операционных систем и не планируется их установка;

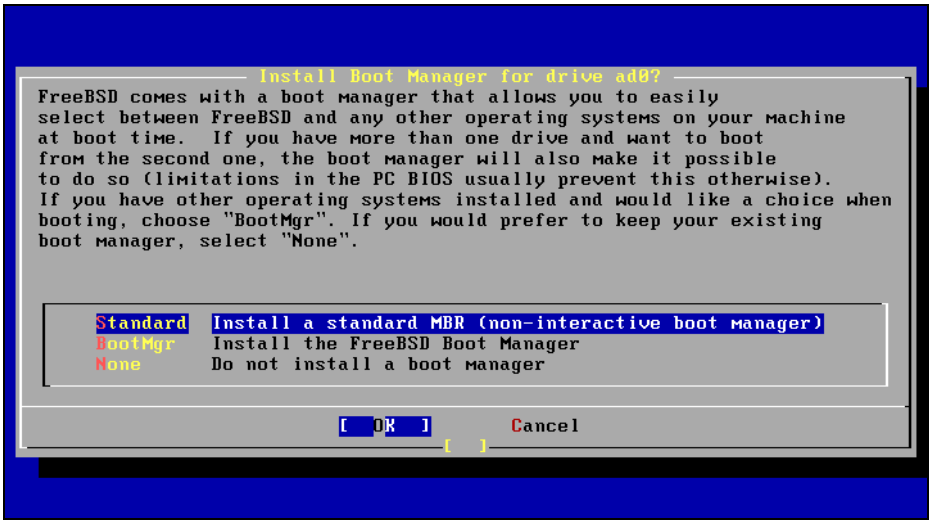


Рис. 2.11. Выбор загрузчика

- **BootMgr** — загрузчик FreeBSD Boot Manager, более продвинутый загрузчик с меню, предлагающим различные варианты загрузки. Он позволяет также организовать загрузку другой операционной системы. И хоть я не планирую установку другой операционной системы, но все же выбрал BootMgr — его возможности шире (мало ли что понадобится в процессе работы с системой);
- **None** — загрузчик вообще не будет установлен, и после установки системы вы не сможете ее загрузить. Такой вариант можно выбрать, если у вас установлена другая ОС (например, Linux), и вы хотите настроить для запуска FreeBSD ее загрузчик.

2.3.5. Создание BSD-разделов внутри BSD-слайса

Настало время создать внутри BSD-слайса BSD-разделы (рис. 2.12). Обратите внимание: система сообщает, что для продолжения установки нужно минимум 1 Гбайт дискового пространства.

Как и при работе со слайсами жесткого диска, вы можете нажать клавишу <A> для автоматического создания BSD-разделов (рис. 2.13). И если в случае со слайсами жесткого диска нужно было соблюдать осторожность — ведь вы могли потерять данные (при их наличии, конечно), то, создавая BSD-разделы, волноваться нечего, можно смело нажимать клавишу <A> для автоматической разметки. Программа установки оптимально распределит имеющееся в ее распоряжении дисковое пространство.

Рассмотрим, как редактор разделов распределил наши 12 Гбайт (рис. 2.14):

- 512 Мбайт — для первого раздела ad0s1a. Этот раздел используется в качестве корневой файловой системы, и поскольку на нем, кроме основных утилит и общесистемных файлов конфигурации, ничего не будет, такого объема более чем достаточно;



Рис. 2.12. Нажмите **OK** для запуска редактора разделов

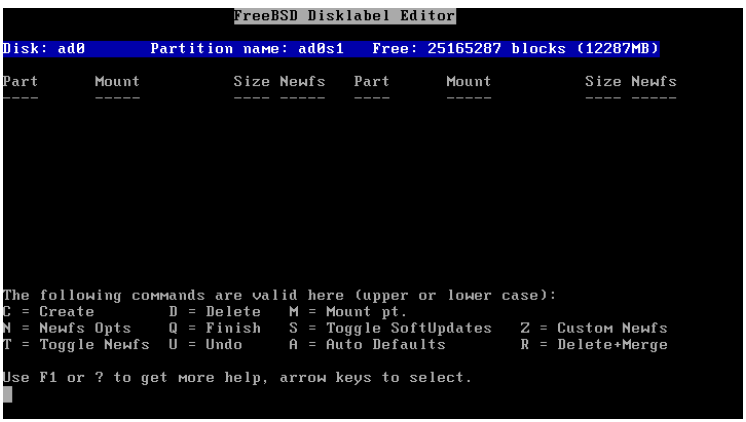


Рис. 2.13. Редактор разделов

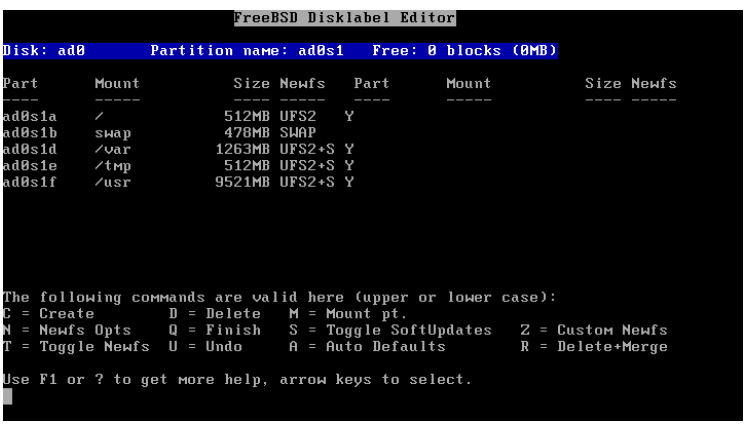


Рис. 2.14. Автоматическое распределение слайса размером 12 Гбайт

- 478 Мбайт — для подкачки (swap), раздел ad0s1b. Учитывая, что FreeBSD хорошо распределяет оперативную память, для подкачки тоже большего объема не нужно. На тестовом компьютере было установлено 256 Мбайт ОЗУ + 478 Мбайт подкачки (всего 734 Майт виртуальной памяти) — этого должно хватить для "среднего" сервера под управлением FreeBSD;
- 1263 Мбайт — для раздела ad0s1d, этот раздел будет монтироваться к каталогу /var. Обычно в этом каталоге хранятся регулярно изменяющиеся данные: например, журналы файловой системы, файлы базы данных СУБД, почтовые ящики пользователей и т. д. Для сервера баз данных 1,2 Гбайт — откровенно мало, но не забываем, что у нас всего 12 Гбайт. Если на диске было бы 120 Гбайт доступного места, тогда для каталога /var было бы отведено 12 Гбайт, а это уже что-то;
- 512 Мбайт — для временных файлов, каталог /tmp, раздел ad0s1e. Тоже должно хватить;
- 9521 Мбайт (все что осталось) — для каталога /usr, раздел ad0s1f. В каталоге /usr находятся установленные программы и необходимые им файлы. 9,5 Гбайт — совсем не много для такого раздела, ведь только коллекция портов, которая будет помещена в каталог /usr/ports, занимает 445 Мбайт.

> ***Примечание***
>
> Напомню, что полное имя раздела формируется так: ad*X*s*N*буква, где *X* — номер диска, *N* — номер слайса. Обратите внимание, какие буквы использованы для имен разделов внутри слайса: a, b, d, e, f. Буква с зарезервирована и никогда не используется.

Итак, нажмите клавишу <A> для автоматического распределения слайса. Посмотрите на созданное разбиение — если вас все устраивает, нажмите клавишу <Q> для продолжения установки. При необходимости распределить место иначе, нажмите клавишу <U> для отмены изменений, а затем <C> — для создания раздела.

Потренируемся в создании разделов. Нажмите клавишу <C>. Редактор разделов попросит указать размер нового раздела. Введите значение `512M` (рис. 2.15).

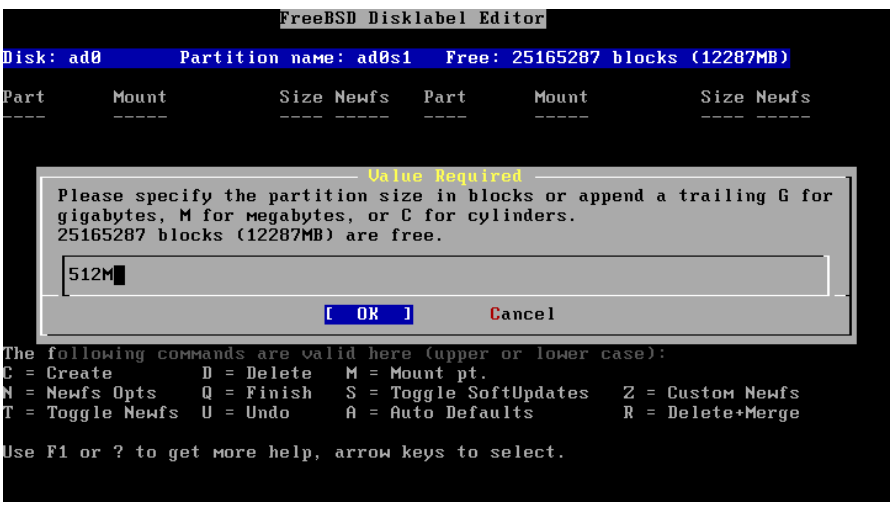


Рис. 2.15. Создание нового раздела

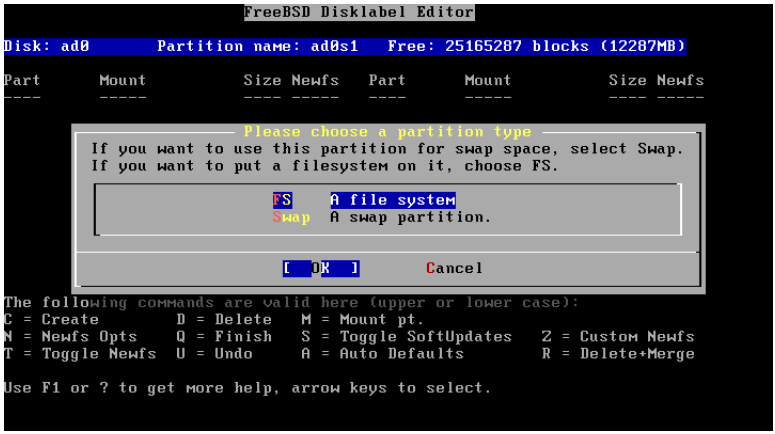


Рис. 2.16. Выбор типа раздела

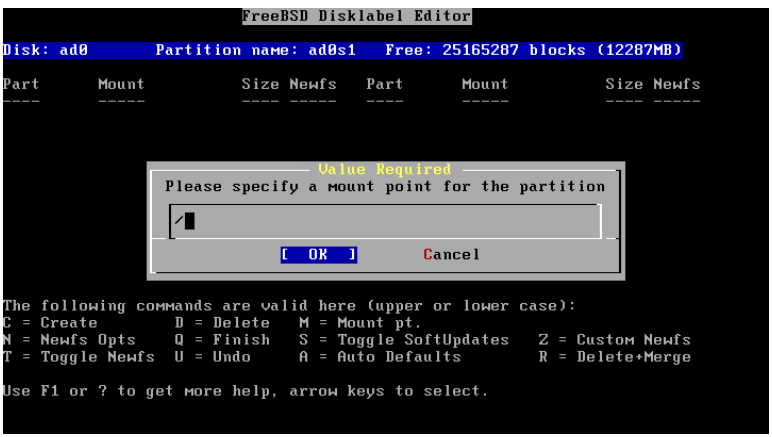


Рис. 2.17. Ввод точки монтирования

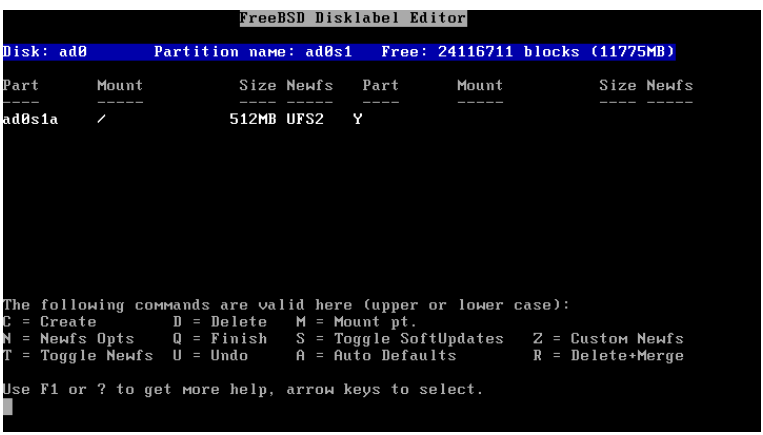


Рис. 2.18. Первый раздел создан

Затем укажите тип раздела: **FS** — для создания файловой системы или **Swap** — для создания раздела подкачки (рис. 2.16). Далее нужно ввести точку монтирования — каталог, к которому будет подмонтирован наш раздел, через него можно будет обратиться к данным, хранящимся на разделе. Введите / для корневой файловой системы (рис. 2.17). Все, первый раздел создан (рис. 2.18).

Теоретически можно создать всего два раздела: первый раздел с емкостью на 512 Мбайт меньше размера всего слайса — для точки монтирования / и второй раздел — раздел подкачки размером 512 Мбайт. Но такая схема распределения пространства в мире UNIX считается дурным тоном. Поэтому, повторив все описанные здесь действия, создайте разделы для точек монтирования /tmp, /var, /usr и не забудьте про раздел подкачки. Размеры разделов установите по своему усмотрению.

2.3.6. Установка программного обеспечения. Выбор источника установки

Теперь вам нужно выбрать дистрибутивный набор программного обеспечения для установки. Варианты предлагаются следующие (рис. 2.19):

- **All** — установить все. Удобно, что все будет под рукой, но займет много места;
- **Reset** — сбросить выбор;
- **Developer** — набор для разработчика: полный набор исходных текстов, документация, но без игр;
- **Kern-Developer** — для разработчика ядра. Будут установлены исходные коды ядра;
- **User** — рассчитанный на среднестатистического пользователя набор программ для рабочей станции;
- **Minimal** — минимальная система. Занимает мало места, но практически ничего не установлено;

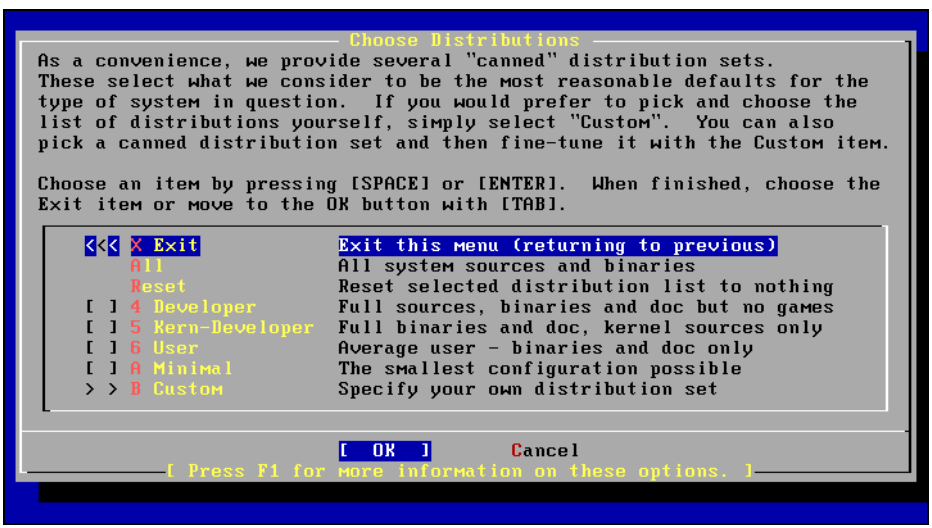


Рис. 2.19. Выбор дистрибутива

- **Custom** — вы сами определяете, что нужно установить. Будьте при этом внимательны: следует обязательно установить дистрибутивы base и kernel (рис. 2.20). Если вы не установите kernel, система вообще не загрузится. Также желательно установить коллекцию портов (ports) и справочную систему (man и info).

Я выбрал вариант **All** — места вполне достаточно. Программа установки спросила, нужно ли устанавливать коллекцию портов (рис. 2.21). Коллекция портов займет 445 Мбайт, но это себя оправдывает. Установка программ из портов будет описана в *главе 18*.



Рис. 2.20. Самостоятельный выбор групп программ

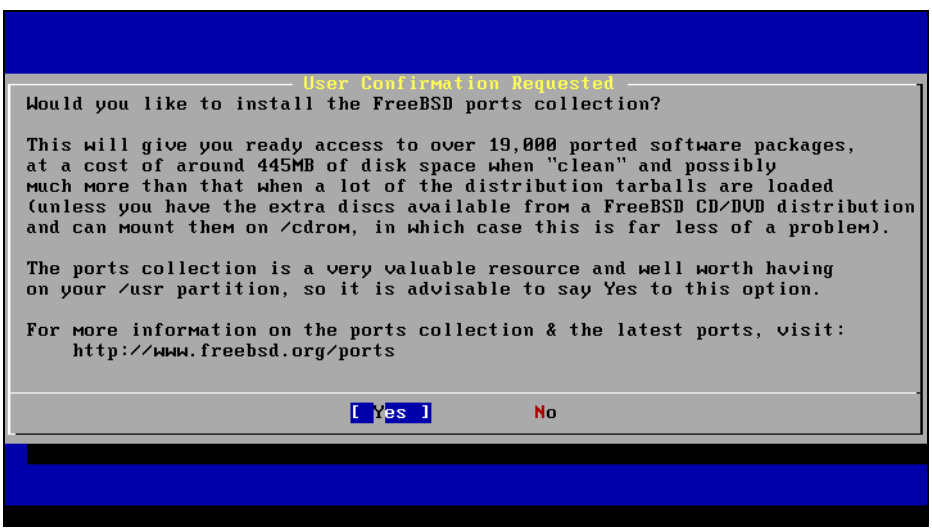


Рис. 2.21. Установить коллекцию портов?

Далее надо выбрать источник установки (рис. 2.22). Выберите вариант **CD/DVD**. Установка по сети, к которой относятся варианты **FTP**, **FTP Passive**, **HTTP**, **NFS**, будет рассмотрена в *главе 7*. Варианты **DOS** и **File system** позволяют установить систему при наличии пакетов на разделе с DOS или на другом разделе с файловой системой BSD. Остальные варианты не интересны. Кстати, вариант **Floppy** — это просто "жесть". Вы можете себе представить установку с набора дискет системы, дистрибутив которой занимает 2,1 Гбайт? — Вставьте, пожалуйста, дискету № 521…

"Вы действительно уверены, что хотите установить FreeBSD?" — Это не я вас спрашиваю, а программа установки (рис. 2.23). Мне бы хотелось посмотреть на человека, который минут двадцать возился с разметкой жесткого диска, а потом передумал устанавливать FreeBSD…

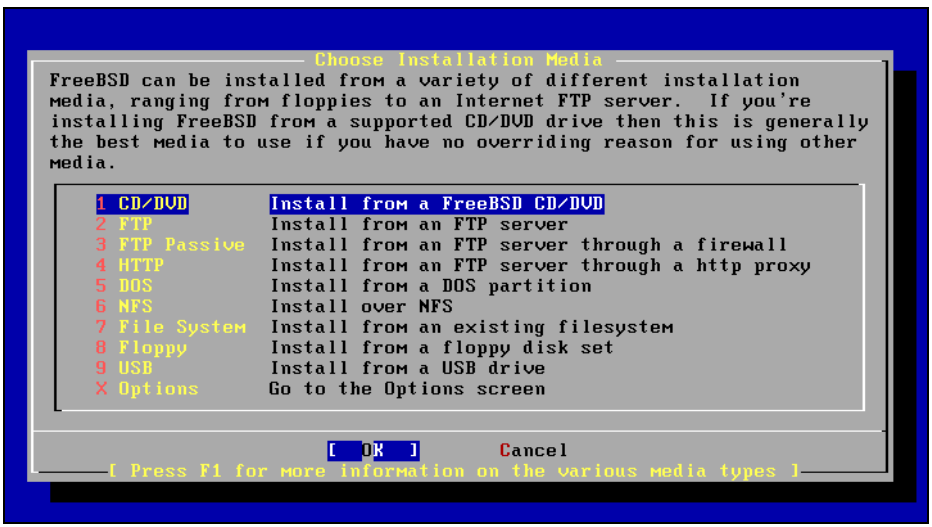


Рис. 2.22. Выбор источника установки

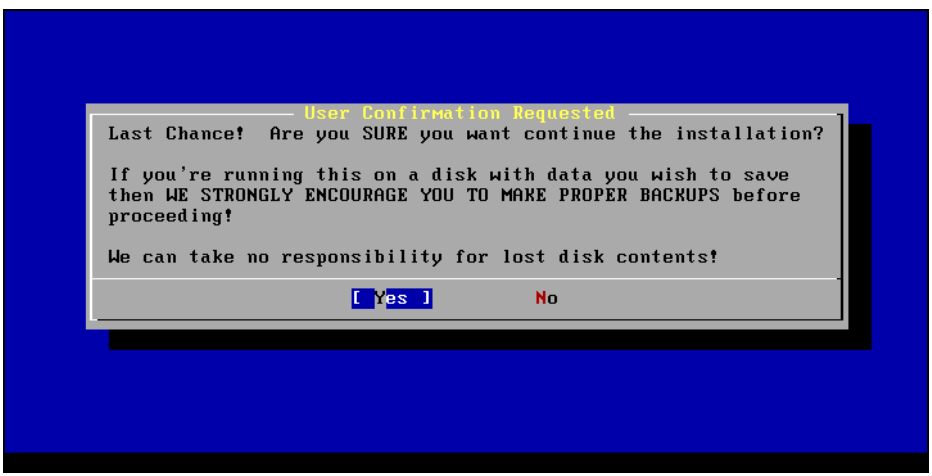


Рис. 2.23. Вы уверены?

Все, осталось только ждать. Если вы выбрали экспресс-установку, система будет перезагружена и вы сможете в нее войти. Имя пользователя — root, пароль отсутствует. А вот если вы выбрали стандартную установку, то после установки самой системы вас ждет постинсталляционная настройка системы.

2.4. Постинсталляционная настройка системы

Через некоторое время система поздравит вас с успешной установкой (рис. 2.24). Далее вам будет задан ряд вопросов по настройке системы. Если вы в чем-то сомневаетесь, просто выберите **No**. Запустить программу настройки вы можете в любой момент после установки системы командой `sysinstall`.

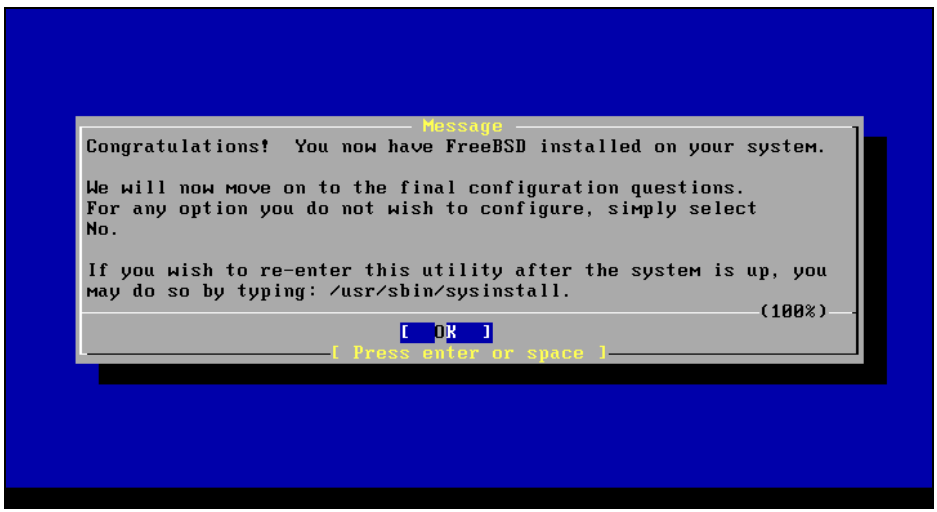


Рис. 2.24. Установка завершена

FreeBSD — это сетевая операционная система, поэтому первым делом у вас спросят, желаете ли вы настроить сеть: **Would you like to configure any Ethernet or SLIP/PPP network devices?** Нужно ответить **Yes**, потому что настройка сети важна (в отличие от дюжины настроек, предлагаемых далее программой установки). Выберите сетевой интерфейс, который хотите настроить. Имейте в виду, что интерфейс **em0** — это как раз сетевая плата (рис. 2.25).

Затем вас спросят, нужно ли включить поддержку IPv6 для этого интерфейса (рис. 2.26). Выберите **No**, поскольку никому не известно, когда будет произведен переход на IPv6 — пока везде используется IPv4.

Использует ли ваша сеть DHCP-сервер (рис. 2.27)? Моя использует — я ответил **Yes**. Это значит, что мой сетевой интерфейс будет настроен автоматически. В случае отсутствия DHCP-сервера выберите **No** — тогда вам придется настроить сетевой интерфейс вручную (рис. 2.28): указать имя узла (**Host**), домен (**Domain**), IP-адрес интерфейса (**IPv4 Address**), IP-адрес шлюза (**IPv4 Gateway**), маску сети (**Netmask**), адрес сервера имен (**Name server**).

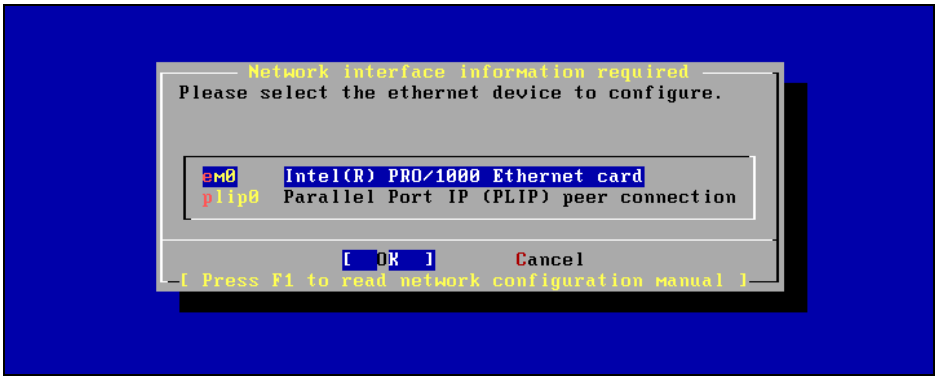


Рис. 2.25. Выбор сетевого интерфейса



Рис. 2.26. Включить поддержку IPv6 для интерфейса?

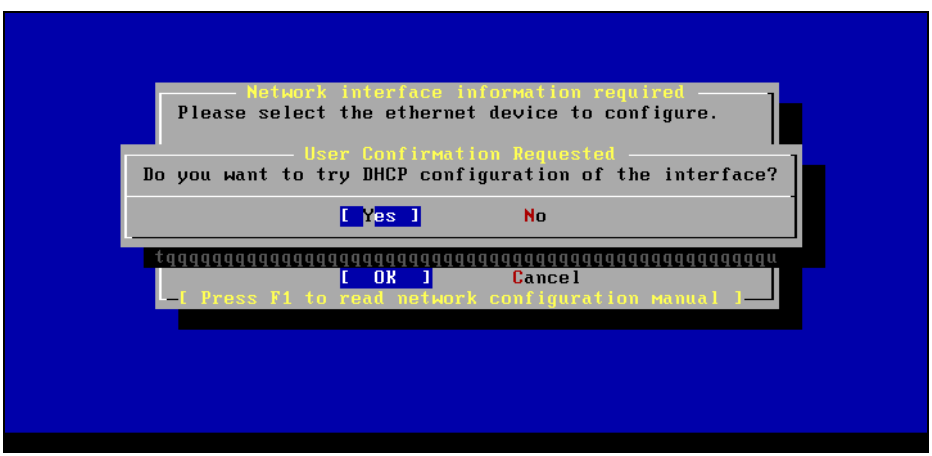


Рис. 2.27. Использует ли сеть DCHP-сервер?

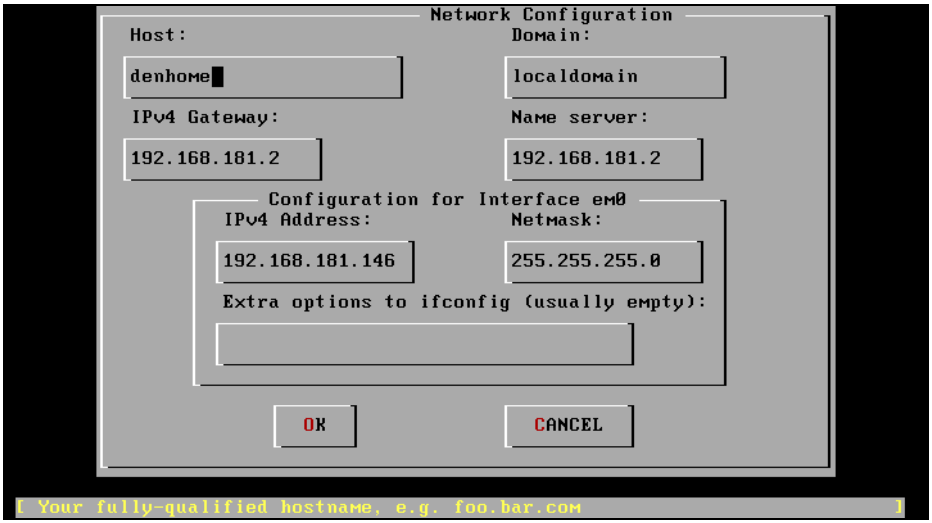


Рис. 2.28. Настройка сети вручную

Далее вам будет задан ряд вопросов относительно настройки сетевых сервисов. Чтобы не приводить в книге бесконечный ряд иллюстраций, я собрал все эти вопросы в табл. 2.1, где пояснил, как нужно отвечать на тот или иной вопрос.

Таблица 2.1. *Вопросы по настройке сетевых сервисов*

Вопрос	Пояснение
Do you want this machine to function as a network gateway? (Должна ли эта машина использоваться в роли сетевого шлюза?)	Пока ответьте **No**, хотя бы потому, что ранее мы указали IP-адрес шлюза, следовательно, один шлюз уже есть. Настройка шлюза рассмотрена в *главе 34*
Do you want to configure inetd and the network services that it provides? (Хотите ли вы настроить сетевые сервисы, запускаемые суперсервером inetd?)	Пока тоже нужно ответить **No**, но к этому вопросу мы еще вернемся в *главе 23* при настройке сети
Would you like to enable SSH login? (Следует ли включить возможность входа в систему по SSH?)	Если вам нужен удаленный доступ прямо сейчас, ответьте **Yes**, а подробно настройку SSH-сервера мы рассмотрим в *главе 33*
Do you want to have anonymous FTP access to this machine? (Включить анонимный FTP-доступ к этой машине?)	Пока этого тоже не стоит делать, а настройка FTP-сервера, в том числе анонимного, будет рассмотрена в *главе 30*
Do you want to configure this machine as an NFS server? (Будет ли эта машина функционировать в роли NFS-сервера/клиента?) **Do you want to configure this machine as an NFS client?** (Вы сможете использовать удаленные файловые системы других серверов?)	О настройке NFS мы поговорим в *главе 31*. Пока, если в вашей сети используется NFS, можно ответить **Yes** на второй вопрос

Следующий вопрос — нужно ли настроить консоль системы (рис. 2.29)? Настраивать здесь пока ничего не нужно по одной простой причине — не стоит сейчас тратить время, все равно стандартным конфигуратором вы не настроите консоль, как нам нужно. Позже мы русифицируем консоль FreeBSD, а с помощью этого конфигуратора русификацию выполнить невозможно.

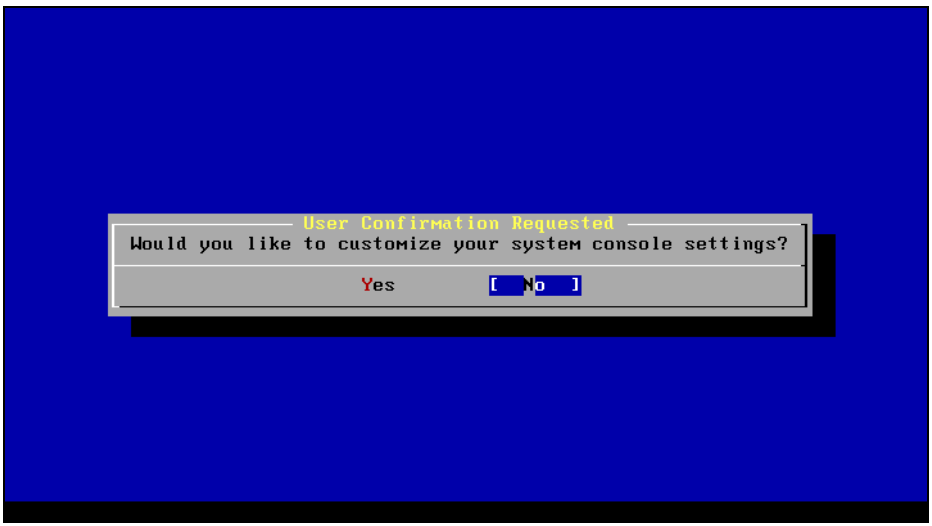


Рис. 2.29. Настраивать консоль не нужно

А вот часовой пояс можно и настроить (рис. 2.30), поэтому выберите **Yes**. На следующий же вопрос (рис. 2.31) следует ответить **No**, поскольку наша система не использует UTC. После этого можно выбрать конкретный часовой пояс (рис. 2.32).

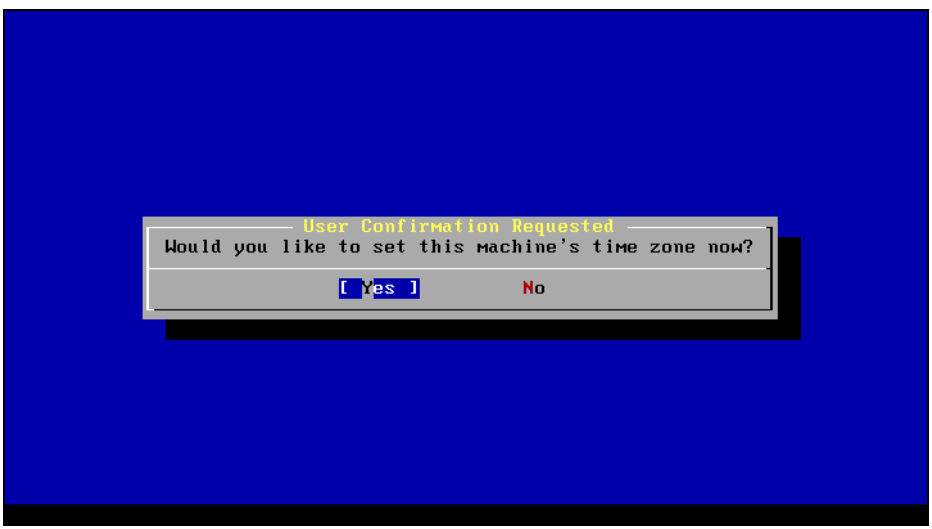
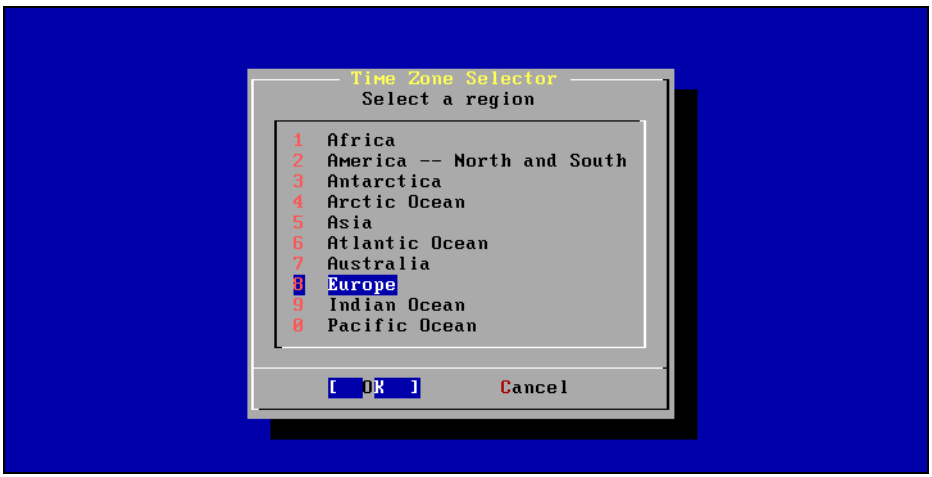


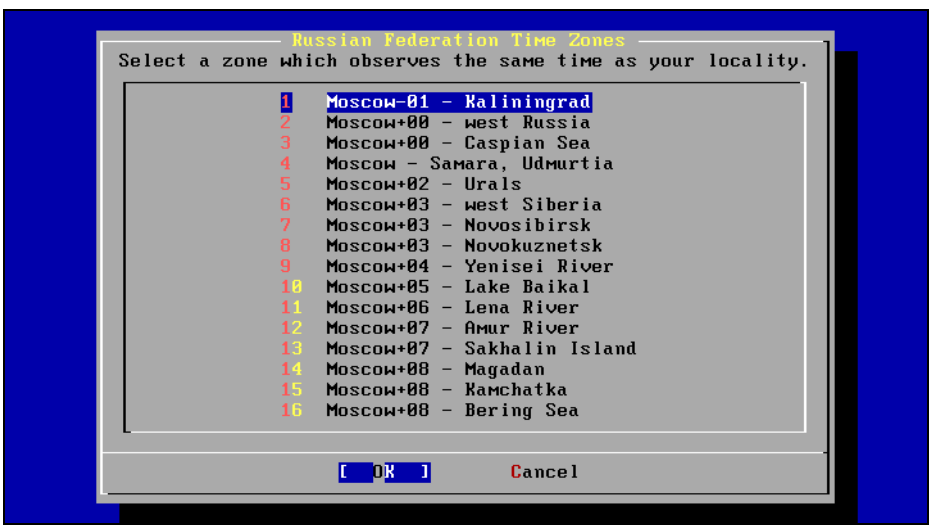
Рис. 2.30. Настроить часовой пояс?



Рис. 2.31. Наша система не использует UTC



а



б

Рис. 2.32. *а* — выбираем регион; *б* — выбираем часовой пояс

От настройки мыши тоже можно отказаться (это следующий вопрос программы установки) — лично мне в консоли мышь не нужна. Но в любом случае, даже если вы сейчас откажетесь настроить мышь, вы всегда сможете сделать это с помощью конфигуратора `sysinstall`.

Не знаю, как вам, а мне уже порядком надоело отвечать то **Yes**, то **No**, хочется начать нормальную работу с системой, а конфигуратор продолжает надоедать своими вопросами. В этот раз — об установке программ (рис. 2.33). Конечно, вы можете просмотреть доступные для установки программы и установить их. Думаю, с выбором программ вы справитесь и без меня (напомню, что установка программ будет подробно рассмотрена в *главе 18*). К этому моменту я пожалел, что не выбрал экспресс-установку.

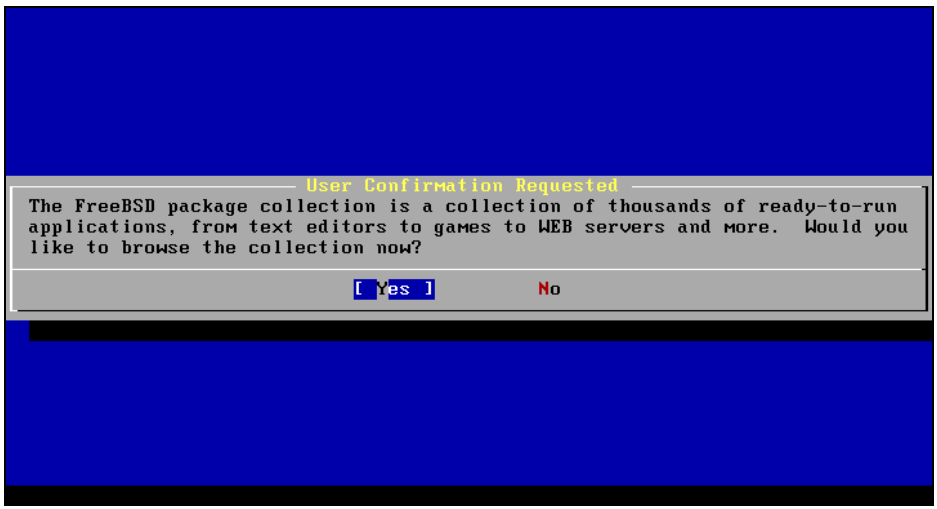


Рис. 2.33. Установить дополнительные программы?

Вот наконец-то конфигуратор задал полезный вопрос — стоит ли добавить в систему пользователей (рис. 2.34)? Выбираем **Yes**, потому что постоянно работать под учетной записью root (суперпользователя, администратора системы) — дурной тон, да и плохо это с точки зрения безопасности. Нет, я сейчас не имею в виду хакеров, а ваши "/dev/hands": одна неправильная и потенциально опасная команда — и систему придется переустанавливать.

На следующем шаге (рис. 2.35) нужно выбрать, что вы хотите добавить: пользователя (**User**) или группу (**Group**). Вообще-то управление пользователями и группами будет рассмотрено в *главе 15*, а пока добавьте одного пользователя (рис. 2.36). При добавлении пользователя следует указать:

- **Login ID** — имя, используемое для входа в систему;
- **UID** — идентификатор пользователя. Устанавливается автоматически, можно не изменять;
- **Group** — группа пользователя, можно не изменять;
- **Password** и **Confirm Password** — пароль и его подтверждение;

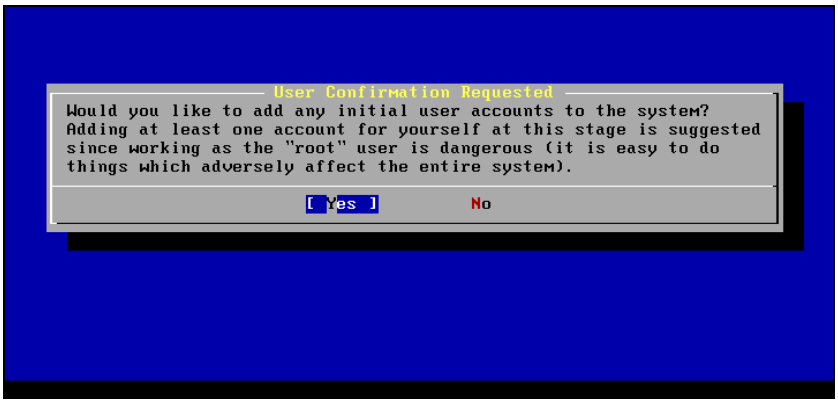


Рис. 2.34. Добавить пользователей?

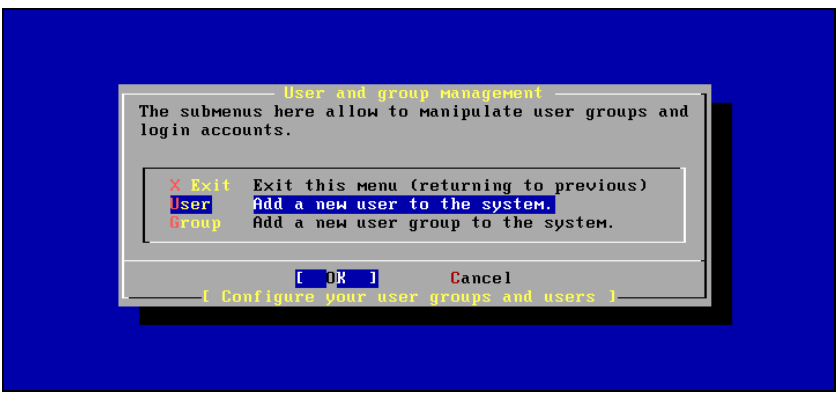


Рис. 2.35. Выберите **User**

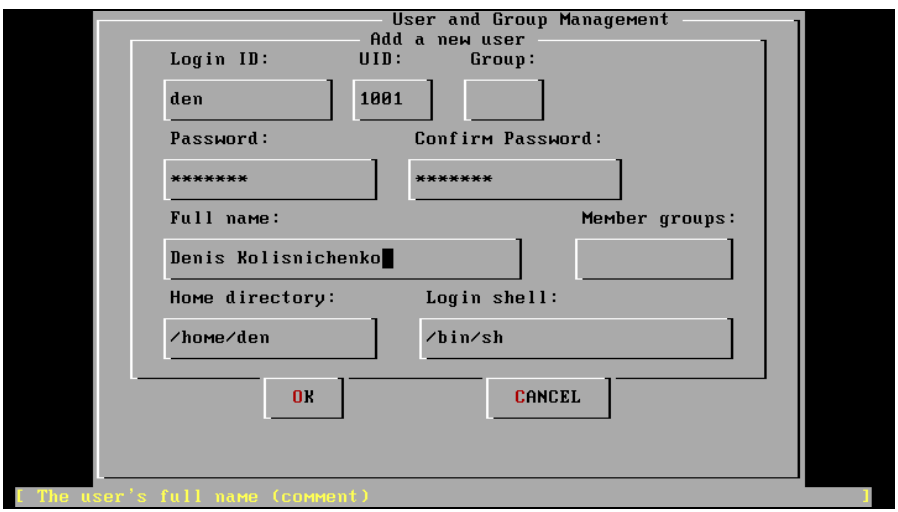
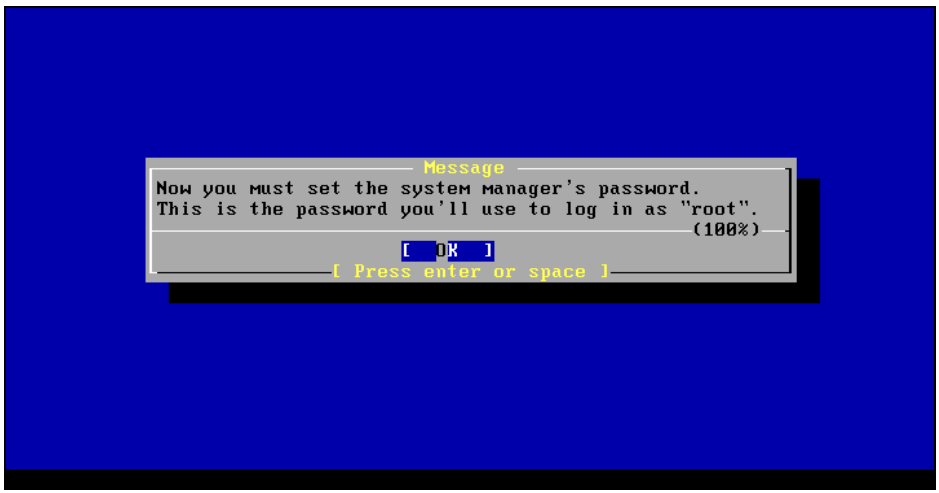


Рис. 2.36. Создание пользователя

- **Full name** — полное имя пользователя;
- **Member groups —** дополнительные группы, в которых участвует пользователь;
- **Home directory** — домашний каталог пользователя;
- **Login shell** — командная оболочка пользователя по умолчанию.

Далее вы должны (да именно должны — отказаться вы не можете) установить пароль root. Ведь по умолчанию у root нет пароля, что очень нехорошо. Сначала нажмите **OK** (рис. 37, *а*), потом введите пароль и его подтверждение. Символы при вводе не отображаются на экране (рис. 2.37, *б*).

Почти все готово. В следующем окне конфигуратор поинтересуется, желаете ли вы вернуться в основное меню, чтобы выполнить дополнительную настройку (рис. 2.38)? Я выбрал ответ **No**, но, тем не менее, почему-то все равно оказался в основном меню (см. рис. 2.5) — пришлось нажать кнопку **Exit install.**



а



б

Рис. 2.37. *а* — нажмите **OK;** *б* — ввод пароля пользователя root

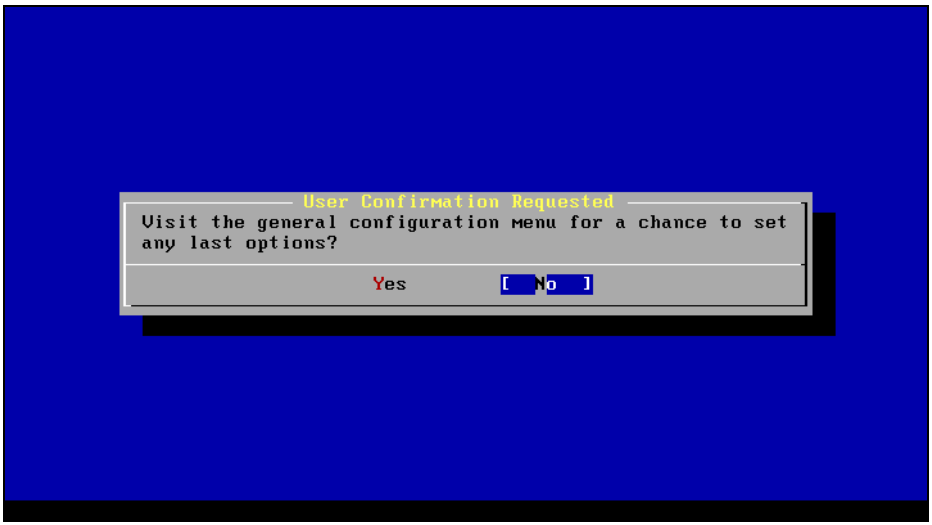


Рис. 2.38. Вернуться в главное меню

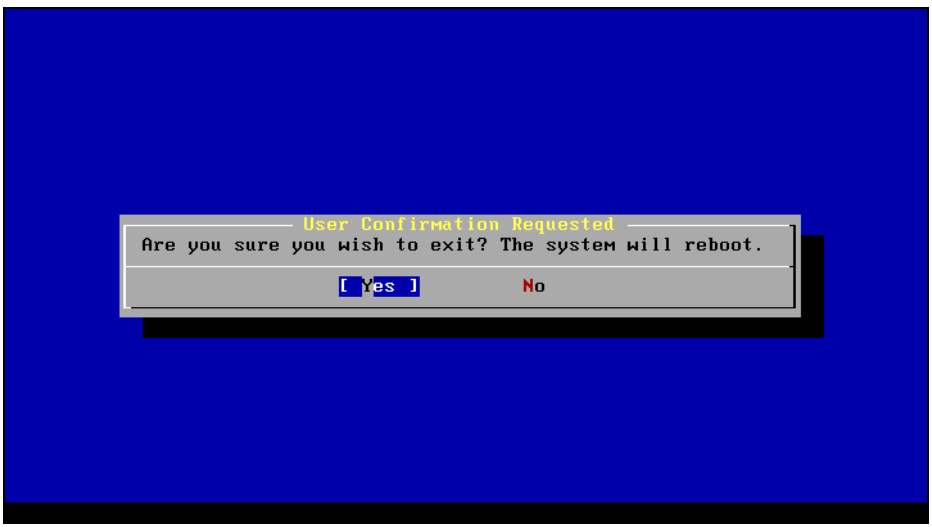


Рис. 2.39. Нажмите **Yes** для перезагрузки

Теперь вас спросят, действительно ли вы хотите выйти, и сообщат, что при этом система будет перезагружена. Наконец-то! Думаю, что вы тоже этого долго ждали (рис. 2.39). Осталось нажать еще раз **OK** в окошке, напоминающем о том, что нужно извлечь инсталляционный диск из привода. Также не забудьте снова установить порядок загрузки — пусть первым делом BIOS ищет загрузчик операционной системы на жестком диске, а не на CDROM.

2.5. После перезагрузки

После перезагрузки для входа в систему вам нужно будет ввести имя пользователя и пароль. Вы можете войти либо как пользователь root, либо как пользователь, созданный при установке системы (рис. 2.40).

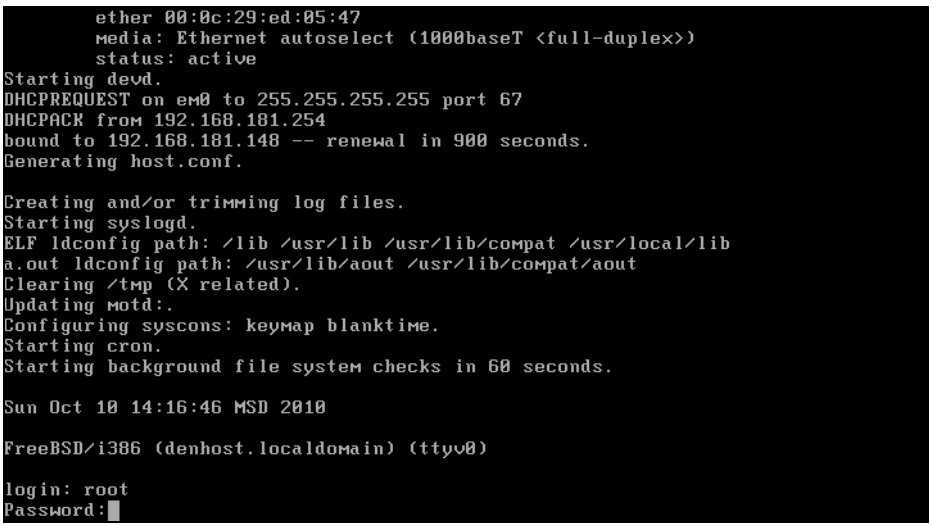


Рис. 2.40. Вход в систему

Для завершения работы необходимо ввести команду `halt` от имени пользователя root. Выключить питание системы можно, как только вы увидите сообщение системы, изображенное на рис. 2.41.

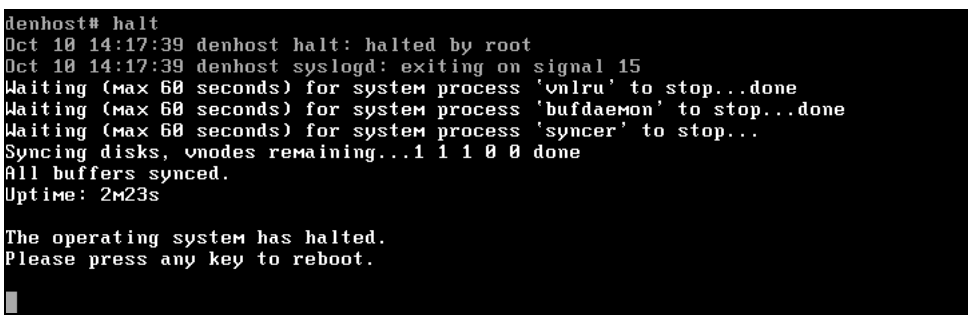


Рис. 2.41. Теперь можно выключить питание компьютера

Для перезагрузки системы надо нажать любую клавишу. Подробнее о входе в систему и о завершении работы мы поговорим в *главе 8*.



Глава 3

Установка OpenBSD

3.1. Перед началом установки

Как уже отмечалось ранее, OpenBSD — еще одна разновидность свободной BSD-совместимой операционной системы. В отличие от других операционных систем, основанных на 4.4BSD (FreeBSD, NetBSD), OpenBSD наиболее безопасная и абсолютно лицензионно чистая.

Согласно данным опроса, проведенного в 2007 году (к сожалению, более свежих данных нет), OpenBSD является второй по популярности BSD-системой. На первом месте — FreeBSD, ее используют около 77% BSD-пользователей, на втором месте OpenBSD с 32% голосов, а на третьем месте — NetBSD (16%).

В этой главе будет рассмотрена установка OpenBSD версии 4.8, которая вышла 1 ноября 2010 года. Это самая последняя версия, и если вы в Интернете нашли руководство по установке OpenBSD, то описание этапов установки в нем будет немного иным — в таких руководствах, как правило, рассматривается версия 4.2 (или 4.3 — в лучшем случае).

Впечатления от установки неоднозначные. С одной стороны, все установилось без проблем, с другой — смотришь на абсолютно текстовую программу установки и думаешь: "Неужели в 2010-м году нельзя было организовать хотя бы псевдографический интерфейс инсталлятора, чтобы процесс установки не выглядел так убого?" После Linux программа установки даже FreeBSD кажется динозавром, не говоря уже про OpenBSD.

Скачать образ дистрибутивного компакт-диска (да, именно CD) можно с одного из FTP-серверов, которые представлены на странице **http://openbsd.org/ftp.html**. Загружать следует файл install48.iso, соответствующий архитектуре процессора вашего компьютера. Например, для архитектуры i386 (устроит большую часть пользователей) заветный файл находится по адресу: **http://ftp.gamma.ru/OpenBSD/4.8/i386/install48.iso**.

Запишите скачанный образ на болванку и загрузитесь с нее. Если на жестком диске, на который вы планируете устанавливать OpenBSD, имеются важные данные, сделайте их резервную копию до начала загрузки инсталлятора.

3.2. Установка системы

3.2.1. Загрузка с компакт-диска

Итак, загрузившись с компакт-диска, вы увидите приглашение загрузчика (рис. 3.1):

boot>

Надо или немного подождать или просто нажать клавишу <Enter>. Начнется процесс загрузки OpenBSD. Все сообщения ядра будут подсвечены синим.

Примечание

Хотел было сделать иллюстрацию, но все равно она получится черно-белой. Просто знайте: все, что подсвечивается синим цветом — это сообщения ядра. После загрузки системы их можно будет найти в файле /var/run/dmesg.boot.

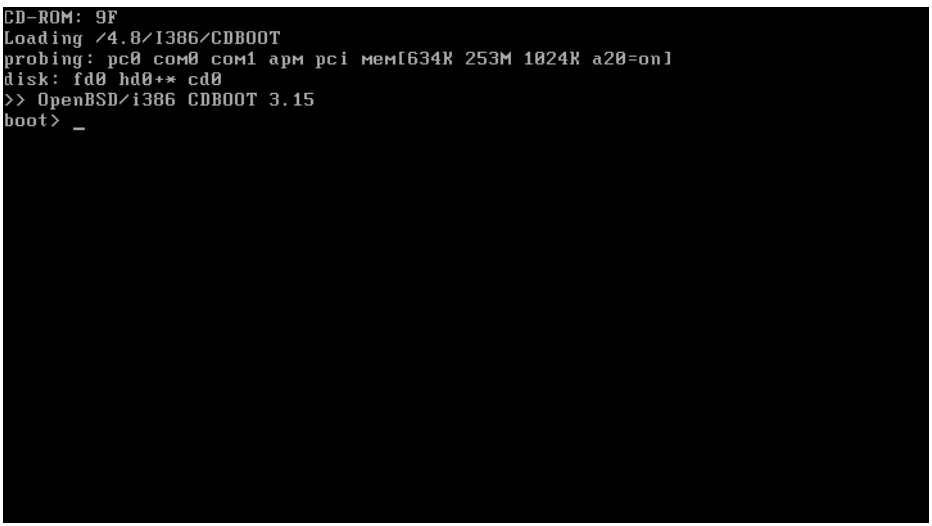


Рис. 3.1. Приглашение загрузчика OpenBSD

Затем вы увидите приглашение программы установки (рис. 3.2):

Welcome to OpenBSD/i386 4.8 installation program

(I)nstall, (U)pgrade or (S)hell? `I`

Поскольку программа установки OpenBSD текстовая, такие вопросы вам будут задаваться постоянно. Никаких окошек с кнопками **OK** и **Cancel** не будет.

Внимание!

Давайте договоримся — вопросы системы в этой книге мы будем выделять **полужирным** шрифтом, а если вопрос требует ответа (как в данном случае), ответ в тексте книги будет выделен шрифтом `Courier`. На большинство вопросов программы установки можно просто ответить нажатием клавиши <Enter>, поскольку значение, предлагаемое по умолчанию (выводится после вопроса в квадратных скобках), практически всегда устроит пользователя.

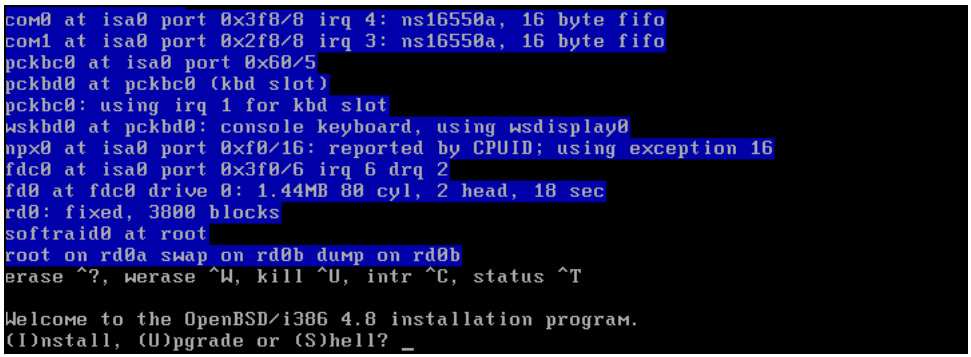


Рис. 3.2. Приглашение инсталлятора OpenBSD

На первый вопрос программы установки следует, как и показано, ввести `i` и нажать клавишу <Enter>, поскольку в оболочку **(S)hell** нам не нужно, а обновлять **(U)pgrade** еще нечего.

3.2.2. Настройка сети

Здесь вам будет задан ряд вопросов (рис. 3.3):

Choose your keyboard layout ('?' or 'L' for list) [default]

System hostname? (short form, e.g. 'foo') `denhost`

Available network interfaces are: em0 vlan0

Which one do you wish to configure? (or 'done') [em0]

IPv4 address for em0? (or 'dhcp' or 'none') [dhcp]

…

IPv6 address for em0? (or 'rtsol' or 'none') [none]

Available network interfaces are: em0 vlan0.

Which one do you wish to configure? (or 'done') [done]

…

Do you want to do any manual network configuration? [no]

На все эти вопросы, кроме второго, я ответил простым нажатием клавиши <Enter>. Но на всякий случай разберемся, что означают все эти вопросы.

Итак, сначала вам предлагают выбрать раскладку клавиатуры. От этого можно отказаться, просто нажав клавишу <Enter> — зачем на сервере русская раскладка? Да и выбрать раскладку при желании можно в любой момент (как заодно и русифицировать всю консоль), хотя бы даже и после установки системы.

Далее нужно ввести имя узла в короткой форме — без домена (я и ввел: `denhost`).

Потом инсталлятор выведет список доступных сетевых интерфейсов:

Available network interfaces are: em0 vlan0

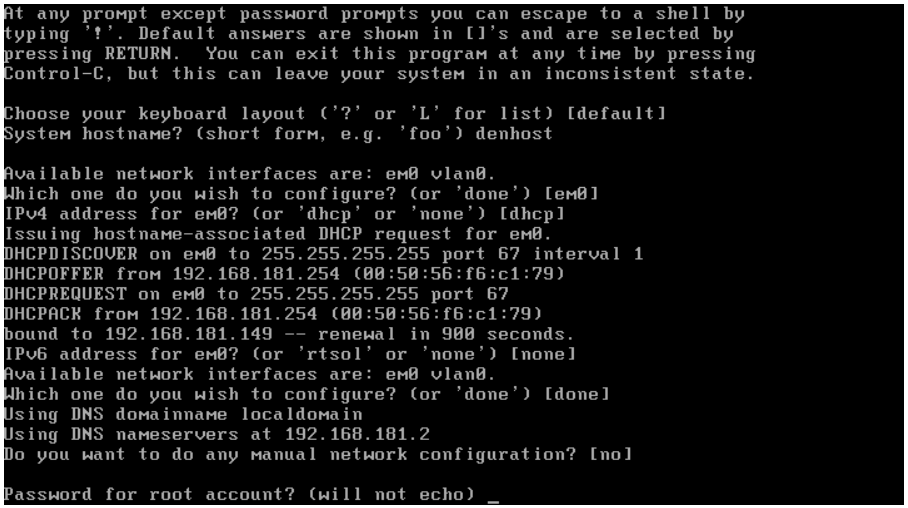


Рис. 3.3. Вопросы инсталлятора до установки пароля root

В ответ нужно ввести имя интерфейса, который вы хотите настроить, или просто нажать клавишу <Enter> для подтверждения предложения инсталлятора. В данном случае он предложил интерфейс **em0** — сетевую карту производства Intel (у вас имя интерфейса может быть другим).

Затем инсталлятор попросит указать IPv4 адрес интерфейса. Если в сети есть DHCP-сервер (как у меня), можно просто нажать клавишу <Enter> для получения сетевых настроек от DHCP. Следующие строки свидетельствуют об успешном получении IP-адреса:

Issuing hostname-associated DHCP request for em0.
DHCPDISCOVER on em0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 192.168.181.254 (00:50:56:f6:c1:79)
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPPACK from 192.168.181.254 (00:50:56:f6:c1:79)
bound to 192.168.181.149 -- renewal in 900 seconds.

Далее нужно указать IPv6-адрес интерфейса или нажать клавишу <Enter> для отказа от ввода адреса. Инсталлятор еще раз выведет список доступных интерфейсов и спросит, какой еще интерфейс вы желаете настроить. На этот раз нажатие клавиши <Enter> означает вариант **done** — завершение настройки сети:

Available network interfaces are: em0 vlan0.
Which one do you wish to configure? (or 'done') [done]

На всякий случай вас спросят (последний вопрос), желаете ли вы настроить сеть вручную? Нажмите клавишу <Enter> для отказа от ручной настройки.

3.2.3. Ввод пароля root

После настройки сети вам будет предложено ввести пароль root (пароль не должен быть пустым, то есть просто так клавишу <Enter> нажать не получится).

При вводе пароль не отображается на экране, после ввода пароля его нужно будет повторить:

Password for root account? (will not echo) `secret`

Password for root account? (again) `secret`

3.2.4. Перед разметкой жесткого диска

Перед разметкой жесткого диска инсталлятор спросит вас, желаете ли вы запускать сервисы sshd и ntpd, планируете ли в будущем запускать графический интерфейс (систему X Window System), будет ли система X Windows запускаться менеджером дисплеев xdm, нужно ли изменить консоль по умолчанию и нужно ли создать обычного пользователя:

Start sshd(8) by default? [yes] `no`

Start nntpd(8) by default? [no]

Do you expect to run the X Window System? [yes]

Do you want the X Window System to be started by xdm(1) [no]

Change the default console to com0? [no]

Setup a user? (center a lower-case loginname, or 'no') [no]

На первый вопрос я ответил отрицательно (`no`), поскольку запускать демон sshd мне не нужно. На остальные вопросы можно ответить нажатием клавиши <Enter>. Последний вопрос — это предложение создать обычного пользователя (рис. 3.4). От этой возможности я отказался, поскольку не вижу смысла тратить время на создание пользователя в процессе установки. Это всегда можно сделать командой `adduser` после установки системы. Если вы таки желаете создать пользователя сразу, введите его имя строчными буквами и нажмите клавишу <Enter>. Затем вам нужно будет ответить на ряд вопросов относительно настройки учетной записи пользователя.

```
IPv4 address for em0? (or 'dhcp' or 'none') [dhcp]
Issuing hostname-associated DHCP request for em0.
DHCPDISCOVER on em0 to 255.255.255.255 port 67 interval 1
DHCPOFFER from 192.168.181.254 (00:50:56:f6:c1:79)
DHCPREQUEST on em0 to 255.255.255.255 port 67
DHCPACK from 192.168.181.254 (00:50:56:f6:c1:79)
bound to 192.168.181.149 -- renewal in 900 seconds.
IPv6 address for em0? (or 'rtsol' or 'none') [none]
Available network interfaces are: em0 vlan0.
Which one do you wish to configure? (or 'done') [done]
Using DNS domainname localdomain
Using DNS nameservers at 192.168.181.2
Do you want to do any manual network configuration? [no]

Password for root account? (will not echo)
Password for root account? (again)
Start sshd(8) by default? [yes] no
Start ntpd(8) by default? [no]
Do you expect to run the X Window System? [yes]
Do you want the X Window System to be started by xdm(1)? [no]
Change the default console to com0? [no]
Setup a user? (enter a lower-case loginname, or 'no') [no]

Available disks are: wd0.
Which one is the root disk? (or 'done') [wd0] _
```

Рис. 3.4. Вопросы до настройки жесткого диска

3.2.5. Разметка жесткого диска

Самый важный этап при установке любой операционной системы — это разметка жесткого диска. Представим, что у нас есть жесткий диск, который можно использовать для установки OpenBSD в полном объеме. По-другому и быть не может — не будете же вы устанавливать на сервер Windows XP?

Инсталлятор выведет список доступных дисков и спросит вас, какой жесткий диск вы желаете разметить (рис. 3.5):

Available disks are: wd0.

Which one is the root disk? (or 'done') [wd0]

Введите имя диска, на который нужно установить OpenBSD, или просто нажмите клавишу <Enter>, если найден всего один жесткий диск.

Внимание!

В OpenBSD диски называются не так, как в FreeBSD. Так, вместо ad*N* (где *N* — число) диски IDE и SATA называются wd*N*, а SCSI-диски и RAID-массивы — sd*N*.

```
Start ntpd(8) by default? [no]
Do you expect to run the X Window System? [yes]
Do you want the X Window System to be started by xdm(1)? [no]
Change the default console to com0? [no]
Setup a user? (enter a lower-case loginname, or 'no') [no]

Available disks are: wd0.
Which one is the root disk? (or 'done') [wd0]
MBR has invalid signature; not showing it.
Use (W)hole disk or (E)dit the MBR? [whole]
Setting OpenBSD MBR partition to whole wd0...done.
The auto-allocated layout for wd0 is:
#                size           offset  fstype [fsize bsize  cpg]
  a:           144.9M               64  4.2BSD   2048 16384    1 # /
  b:           144.9M           296736    swap
  c:          8192.0M                0  unused
  d:           223.8M           593440  4.2BSD   2048 16384    1 # /tmp
  e:           248.6M          1051712  4.2BSD   2048 16384    1 # /var
  f:           964.9M          1560928  4.2BSD   2048 16384    1 # /usr
  g:           550.9M          3536960  4.2BSD   2048 16384    1 # /usr/X11R6
  h:          2177.7M          4665216  4.2BSD   2048 16384    1 # /usr/local
  i:          1062.9M          9125184  4.2BSD   2048 16384    1 # /usr/src
  j:          1062.9M         11302016  4.2BSD   2048 16384    1 # /usr/obj
  k:          1607.8M         13478848  4.2BSD   2048 16384    1 # /home
Use (A)uto layout, (E)dit auto layout, or create (C)ustom layout? [a] _
```

Рис. 3.5. Процесс разметки жесткого диска

Далее вас спросят, нужно использовать весь диск (**Whole**) или вы предпочитаете редактировать таблицу разделов вручную (**Edit**):

Use (W)hole disk or (E)dit the MBR? [whole]

Если на вашем жестком диске имеется хотя бы один раздел с данными плюс нераспределенное пространство, и при этом вы не хотите потерять данные, выберите вариант E — будет запущена программа fdisk, с помощью которой вы сможете разметить нераспределенное пространство диска вручную. Если же нераспределенного пространства нет, не надейтесь, что fdisk умеет изменять размер раздела без потери данных. Вам придется удалить какой-то раздел и создать раздел для OpenBSD на его месте.

Позволю себе небольшой совет: подключите жесткий диск, на который вы планируете установить OpenBSD, к другому компьютеру, скопируйте все важные данные, а потом выберите вариант `W` (использование всего жесткого диска) — так намного проще. Будет создан один слайс (привыкайте к терминологии BSD) на весь жесткий диск, после чего следует создать BSD-разделы внутри этого слайса. Надеюсь, вы прочитали *главу 2*, где рассматривалась установка FreeBSD? Если нет, тогда сделайте это прямо сейчас — вам будет понятна организация дискового пространства в BSD.

После выполнения указанных процедур, инсталлятор выведет разметку слайса по умолчанию (см. рис. 3.5). Вы можете принять автоматическую разметку, отредактировать ее или создать собственную разметку:

Use (A)uto layout, (E)dit auto layout, or create (C)ustom layout? [a]

Честно говоря, мне автоматическая разметка (см. рис. 3.5) не понравилась. Во-первых, для каталога /var отводится слишком мало места, а в этом каталоге обычно хранятся и почтовые ящики, и файлы Web-сервера, и базы данных, и кэш прокси-сервера. 248 Мбайт, а именно столько в данном случае было отведено, под все эти нужды маловато. Во-вторых, для каталога /usr и его подкаталогов создано аж пять разделов — слишком много. Я предлагаю для каталога /var выделить больше места — как минимум 2 Гбайт, а для подкаталогов каталога /usr не создавать отдельных разделов.

Если вы спешите, можно, конечно, принять и автоматическую разметку, нажав клавишу <Enter>. Но потом вы будете с ней мучиться — попомните тогда мои слова! Поэтому лучше выберите вариант `C` для разметки вручную.

Опять-таки, если нет желания создавать разделы и планировать дисковое пространство, можете создать только разделы a и b. Эти разделы обязательны, и без них продолжить установку нельзя. Раздел a будет монтироваться к корневому каталогу (/), его размер будет равен размеру диска минус размер раздела b. Раздел b — это раздел подкачки. На современных компьютерах достаточно памяти, поэтому можете не злоупотреблять размером этого раздела: 256–512 Мбайт вполне достаточно. На загруженных серверах рекомендуют установить размер подкачки, в 2 раза превышающий размер оперативной памяти. Если вы планируете создать именно такой сервер, и ваш компьютер оснащен ОЗУ размером 1 Гбайт, размер раздела b должен быть 2 Гбайт.

Итак, мы выбрали команду `C`, в результате ее выполнения будет запущен редактор разделов (disklabel):

Initial label editor (enter '?' for help at any promt)

>

Можно ввести команду `?` для получения справки, но я вам и так расскажу, что к чему. Команда `a` создает раздел. В качестве аргумента ей нужно передать имя раздела:

> `a a`

offset: [63] — нажмите клавишу <Enter>

size: [20474317] `256m`

Rounding to nearest cylinder: 530082

FS type: [4.2BSD] — нажмите клавишу <Enter>

mount point: [none] `/`

В этой записи мы при создании раздела указываем смещение от начала раздела (**offset**) — обычно достаточно просто нажать клавишу <Enter>, задаем размер раздела (можно указать `m`, чтобы задать раздел в мегабайтах), задаем тип файловой системы (тоже нажатием клавиши <Enter>) и точку монтирования. В результате создается разметка, описанная в табл. 3.1, — чтобы вы представляли, что делаете.

Таблица 3.1. *Оптимальная разметка BSD-слайса*

Раздел	Размер	Точка монтирования	Описание
a	256 Мбайт	/	Для корневой файловой системы
b	512 Мбайт	—	Раздел подкачки
c	—	—	Раздел с не используется, он зарезервирован для служебных целей
d	от 4 Гбайт	/usr	В этот каталог будут устанавливаться программы, поэтому его размер зависит от того, что вы планируете установить. Не жадничайте — на современных жестких дисках достаточно места
e	от 2 Гбайт	/var	Здесь хранятся почтовые ящики пользователей, кэш прокси-сервера, файлы Web-сервера, файлы анонимного FTP-сервера. Размер этого раздела зависит от того, какой сервер вы планируете настроить
f	от 2 Гбайт	/home	Пользовательские каталоги

Если места на диске мало, то можно сэкономить на разделе f — пользовательским каталогам нужно очень мало места, особенно, если не предусматривается хранение на сервере файлов пользователей. Поэтому для раздела f можете смело отвести 256 Мбайт, а сэкономленное место отдать разделу e.

Обратите внимание, что при создании раздела b автоматически предлагается тип раздела swap — подкачка, вам даже ничего не придется выбирать, а нужно просто нажать клавишу <Enter>:

```
> a b
```

offset: [530145] — нажмите клавишу <Enter>

size: [19888470] `512m`

Rounding to nearest cylinder: 1060290

FS type: [swap] — нажмите клавишу <Enter>

Для выхода из программы разметки введите команду `q`, а затем нажмите клавишу <Enter>:

```
> q
```

Write new label?: [y]

3.2.6. Дистрибутивные наборы

После разметки диска инсталлятор спросит вас, откуда загружать дистрибутивные наборы:

Location of sets? (cd disk ftp http or 'done') [cd]

По умолчанию подразумевается установка с компакт-диска — просто нажмите клавишу <Enter>. Далее придется еще раз нажать клавишу <Enter> (рис. 3.6) — когда вас спросят, с какого именно CD-привода нужно получить инсталляционные наборы, и когда попросят указать путь к этим наборам — обычно есть всего один привод cd0, его и нужно выбрать нажатием клавиши <Enter>.

```
Let's install the sets!
Location of sets? (cd disk ftp http or 'done') [cd]
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0]
Pathname to the sets? (or 'done') [4.8/i386] _
```

Рис. 3.6. Откуда устанавливать программное обеспечение?

Следующий шаг — выбор дистрибутивных наборов (рис. 3.7):

- bsd — это ядро системы, данный набор жизненно необходим, не выключайте его;
- bsd.rd — версия ядра с RAM-диском;
- bsd.mp — версия ядра для многопроцессорных машин (SMP);
- base48.tgz — базовая система OpenBSD (обязательный набор);
- etc48.tgz — все конфигурационные файлы из каталога /etc (обязательный набор);
- misc48.tgz — информационные материалы, документация (необязательный набор);

```
5 cylinder groups of 61.91MB, 3962 blocks, 7936 inodes each
/dev/wd0a on /mnt type ffs (rw, asynchronous, local)
/dev/wd0k on /mnt/home type ffs (rw, asynchronous, local, nodev, nosuid)
/dev/wd0d on /mnt/tmp type ffs (rw, asynchronous, local, nodev, nosuid)
/dev/wd0f on /mnt/usr type ffs (rw, asynchronous, local, nodev)
/dev/wd0g on /mnt/usr/X11R6 type ffs (rw, asynchronous, local, nodev)
/dev/wd0h on /mnt/usr/local type ffs (rw, asynchronous, local, nodev)
/dev/wd0j on /mnt/usr/obj type ffs (rw, asynchronous, local, nodev, nosuid)
/dev/wd0i on /mnt/usr/src type ffs (rw, asynchronous, local, nodev, nosuid)
/dev/wd0e on /mnt/var type ffs (rw, asynchronous, local, nodev, nosuid)

Let's install the sets!
Location of sets? (cd disk ftp http or 'done') [cd]
Available CD-ROMs are: cd0.
Which one contains the install media? (or 'done') [cd0]
Pathname to the sets? (or 'done') [4.8/i386]

Select sets by entering a set name, a file name pattern or 'all'. De-select
sets by prepending a '-' to the set name, file name pattern or 'all'. Selected
sets are labelled '[X]'.
    [X] bsd           [X] etc48.tgz     [X] game48.tgz    [X] xfont48.tgz
    [X] bsd.rd        [X] misc48.tgz    [X] xbase48.tgz   [X] xserv48.tgz
    [ ] bsd.mp        [X] comp48.tgz    [X] xetc48.tgz
    [X] base48.tgz    [X] man48.tgz     [X] xshare48.tgz
Set name(s)? (or 'abort' or 'done') [done] _
```

Рис. 3.7. Выбор дистрибутивных наборов

- comp48.tgz — компиляторы, заголовочные файлы и библиотеки (рекомендуемый набор);
- man48.tgz — справочная система (рекомендуемый набор);
- game48.tgz — игры (необязательный набор);
- xbase48.tgz — базовая установка X11 — X Windows System (необязательный набор);
- xetc48.tgz — конфигурационные файлы /etc/X11 и /etc/fonts (необязательный набор);
- xshare48.tgz — вспомогательные файлы для X11 (необязательный набор);
- xfont48.tgz — сервер шрифтов и шрифты для X11 (необязательный набор);
- xserv48.tgz — серверы X11.

Чтобы включить дистрибутивный набор, введите его имя, для отключения дистрибутивного набора введите выражение `-имя`. Например, для выключения набора bsd.rd нужно ввести `-bsd.rd`. Для выбора всех наборов введите `all`. Когда завершите выбор наборов, введите `done`.

Далее начнется процесс получения наборов (рис. 3.8). Придется немного подождать, пока наборы будут скопированы с компакт-диска. После этого инсталлятор снова попросит указать источник наборов:

Location of sets? (cd disk ftp http or 'done') [cd]

Поскольку дополнительных источников наборов у нас нет, просто нажмите клавишу <Enter> для завершения этого этапа.

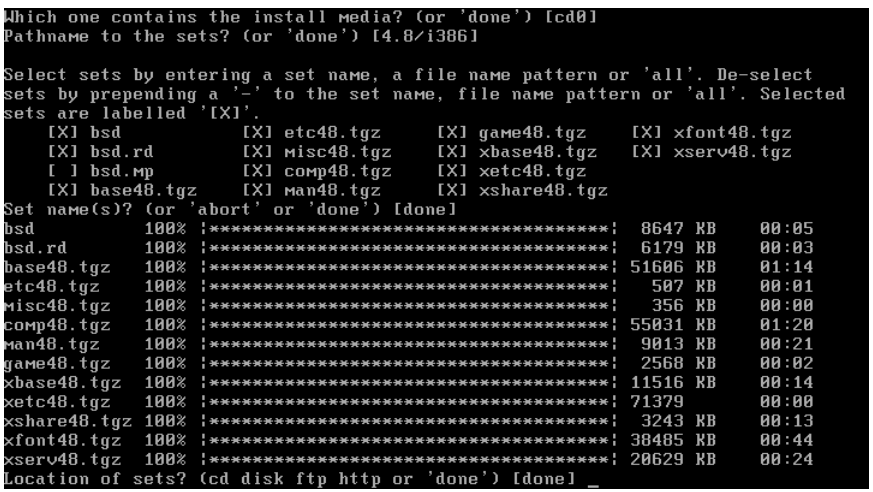


Рис. 3.8. Получение дистрибутивных наборов

3.2.7. Выбор часового пояса

Пришло время выбрать часовой пояс:

What timezone are you in? ('?' for list) [Canada/Mountain] `?`

Введите `?` для просмотра доступных вариантов (рис. 3.9), хотя можно сразу ввести `Europe`, а уже затем ввести `?` для уточнения вашего местонахождения.

```
What timezone are you in? ('?' for list) [Canada/Mountain] ?
Africa/       Chile/        GB-Eire       Israel        NZ-CHAT       UCT
America/      Cuba          GMT           Jamaica       Navajo        US/
Antarctica/   EET           GMT+0         Japan         PRC           UTC
Arctic/       EST           GMT-0         Kwajalein     PST8PDT       Universal
Asia/         EST5EDT       GMT0          Libya         Pacific/      W-SU
Atlantic/     Egypt         Greenwich     MET           Poland        WET
Australia/    Eire          HST           MST           Portugal      Zulu
Brazil/       Etc/          Hongkong      MST7MDT       ROC           posix/
CET           Europe/       Iceland       Mexico/       ROK           posixrules
CST6CDT       Factory       Indian/       Mideast/      Singapore     right/
Canada/       GB            Iran          NZ            Turkey
What timezone are you in? ('?' for list) [Canada/Mountain] Europe
What sub-timezone of 'Europe' are you in? ('?' for list) ?
Amsterdam     Chisinau      Kiev          Moscow        Sarajevo      Vatican
Andorra       Copenhagen    Lisbon        Nicosia       Simferopol    Vienna
Athens        Dublin        Ljubljana     Oslo          Skopje        Vilnius
Belfast       Gibraltar     London        Paris         Sofia         Volgograd
Belgrade      Guernsey      Luxembourg    Podgorica     Stockholm     Warsaw
Berlin        Helsinki      Madrid        Prague        Tallinn       Zagreb
Bratislava    Isle_of_Man   Malta         Riga          Tirane        Zaporozhye
Brussels      Istanbul      Mariehamn     Rome          Tiraspol      Zurich
Bucharest     Jersey        Minsk         Samara        Uzhgorod
Budapest      Kaliningrad   Monaco        San_Marino    Vaduz
What sub-timezone of 'Europe' are you in? ('?' for list) _
```

Рис. 3.9. Установка часового пояса

3.2.8. Перезагрузка системы

После выбора часового пояса нужно нажать клавишу <Enter> для подтверждения текущего времени, после чего вы увидите долгожданное поздравление с установкой системы (рис. 3.10).

```
Canada/       GB            Iran          NZ            Turkey
What timezone are you in? ('?' for list) [Canada/Mountain] Europe
What sub-timezone of 'Europe' are you in? ('?' for list) ?
Amsterdam     Chisinau      Kiev          Moscow        Sarajevo      Vatican
Andorra       Copenhagen    Lisbon        Nicosia       Simferopol    Vienna
Athens        Dublin        Ljubljana     Oslo          Skopje        Vilnius
Belfast       Gibraltar     London        Paris         Sofia         Volgograd
Belgrade      Guernsey      Luxembourg    Podgorica     Stockholm     Warsaw
Berlin        Helsinki      Madrid        Prague        Tallinn       Zagreb
Bratislava    Isle_of_Man   Malta         Riga          Tirane        Zaporozhye
Brussels      Istanbul      Mariehamn     Rome          Tiraspol      Zurich
Bucharest     Jersey        Minsk         Samara        Uzhgorod
Budapest      Kaliningrad   Monaco        San_Marino    Vaduz
What sub-timezone of 'Europe' are you in? ('?' for list) Kiev
Time appears wrong.  Set to 'Sat Nov 27 12:29:20 EET 2010'? [yes]
Saving configuration files...done.
Generating initial host.random file...done.
Making all device nodes...done.

CONGRATULATIONS! Your OpenBSD install has been successfully completed!
To boot the new system, enter 'reboot' at the command prompt.
When you login to your new system the first time, please read your mail
using the 'mail' command.

# _
```

Рис. 3.10. Установка завершена

Введите команду `reboot` для перезагрузки системы. После перезагрузки вы можете войти в систему, используя имя пользователя root и пароль, указанный при установке (рис. 3.11).

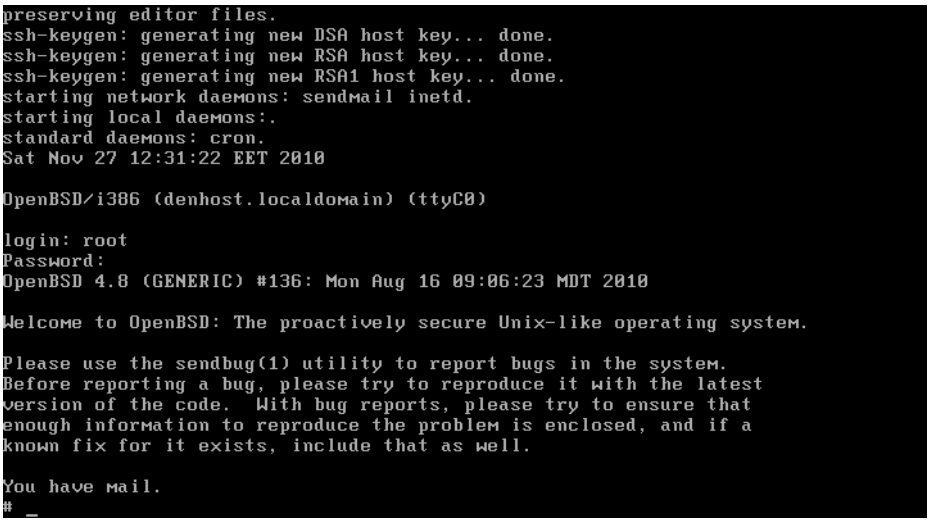


Рис. 3.11. Вход в систему выполнен



Глава 4

Операционная система РУС-BSD: обзор, установка

4.1. Кратко о системе

Операционная система РУС-BSD основана на FreeBSD, но обладает полностью иным, более удобным графическим инсталлятором, Windows-подобным графическим интерфейсом пользователя и набором протестированных пользовательских программ. Звук и видео работают в РУС-BSD "из коробки", то есть сразу после установки вы получаете полностью настроенную систему. Все это позволяет использовать РУС-BSD не только на сервере, но и на рабочей станции. Не спорю, можно и FreeBSD настроить для работы на рабочей станции — проект РУС-BSD тому подтверждение, но на подобную настройку уйдет намного больше времени, нежели на установку РУС-BSD.

РУС-BSD — это совместный проект русских, украинских, белорусских и казахстанских разработчиков, поэтому с поддержкой родных для них языков проблем не будет. В перспективе ожидается локализация проекта для других стран бывшего СССР.

Если вы фанат FreeBSD и вам не хотелось устанавливать Linux на рабочие станции своей сети, то РУС-BSD для вас — выход. Пользователи РУС-BSD смогут работать в Интернете, совершать звонки по Skype, работать с офисными приложениями и электронной почтой — и все это без напильника и танца с бубном вокруг компьютера.

Бесплатно скачать РУС-BSD можно с сайта разработчиков: **http://rusbsd.org/**. На момент написания этих строк доступна версия 8.0.1 "Брест", основанная на версии FreeBSD 8.1.

Минимальные системные требования РУС-BSD следующие:

- Процессор — не ниже Pentium-II;
- Оперативная память — 128 Мбайт (при использовании файловой системы UFS2) или 1024 Мбайт (для файловой системы ZFS);
- Дисковое пространство — 4 Гбайт.

Рекомендуемые системные требования:

- Частота процессора — не ниже 1 ГГц;
- Оперативная память — 512 Мбайт (UFS2) или 1,5 Гбайт (ZFS);
- Дисковое пространство — от 12 Гбайт (для полной установки всех компонентов + место для пользовательских данных).

Да, PYC-BSD немного прожорлива, но все компоненты вы вряд ли будете устанавливать, а ZFS пока использовать не рекомендуется — она еще экспериментальная, да и также довольно прожорлива.

4.2. Установка PYC-BSD

Подробно рассматривать установку и настройку PYC-BSD в этой книге мы не будем. Тому есть несколько причин. Первая — очень простой графический и русскоязычный инсталлятор. Установка PYC-BSD не сложнее установки Linux или даже Windows — с ней справится даже школьник (учитывая автоматическую разметку диска). Да, в мире BSD все из крайности в крайность: установка OpenBSD — слишком сложна, установка PYC-BSD — слишком проста. FreeBSD на фоне этих двух операционных систем кажется "золотой серединой" — и не слишком сложно, и не слишком просто.

Вторая причина в том, что PYC-BSD основана на FreeBSD — все конфигурационные файлы, команды будут такими же. Можете наслаждаться работой в PYC-BSD и продолжать читать эту книгу, не обращая внимания на слово "FreeBSD": все, что будет сказано для FreeBSD, верно и для PYC-BSD.

Итак, приступим к установке PYC-BSD. Первое, что бросается в глаза после загрузки с диска — меню загрузчика на "почти русском языке", в транслитерации (рис. 4.1). Нажмите клавишу <Enter> для выбора первого варианта: **Zagruzka**. Начнется процесс загрузки, который в PYC-BSD частично русифицирован (рис. 4.2).



Рис. 4.1. Меню загрузчика

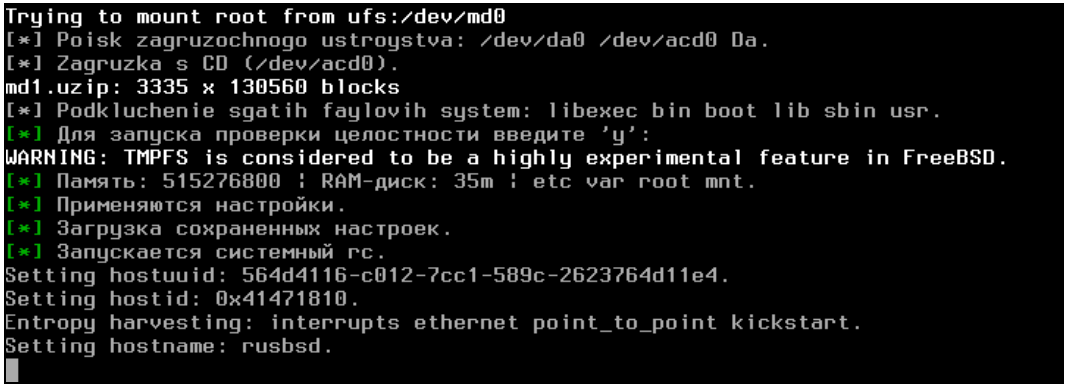


Рис. 4.2. Процесс загрузки РУС-BSD

Далее откроется меню запуска программы установки. По умолчанию инсталлятор запускается в графическом режиме с максимальным поддерживаемым разрешением, но вы можете выбрать меньшее разрешение, например, 800×600 (рис. 4.3).

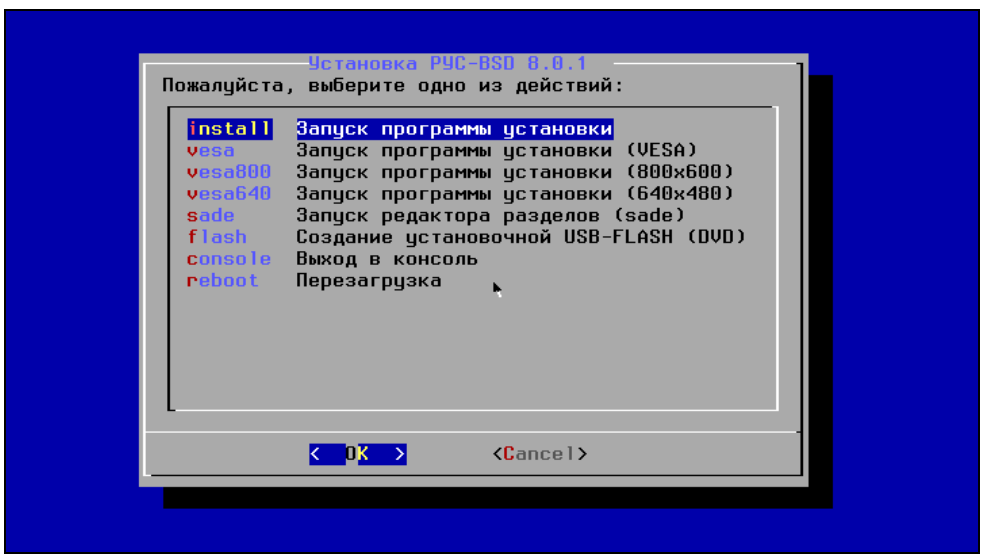


Рис. 4.3. Меню запуска инсталлятора

Последующий процесс установки напоминает Linux или даже Windows: выбор раскладки, часового пояса, чтение лицензионного соглашения (рис. 4.4), выбор типа установки (рис. 4.5).

С дальнейшей установкой вы справитесь сами — даже не знаю, что может произойти, чтобы вы не смогли установить РУС-BSD.

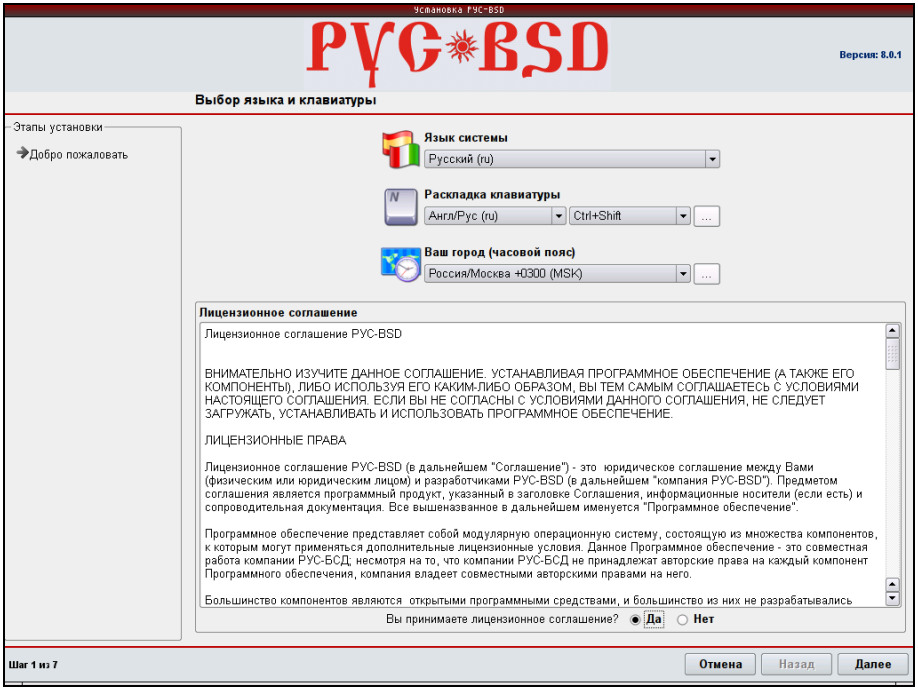


Рис. 4.4. Лицензионное соглашение

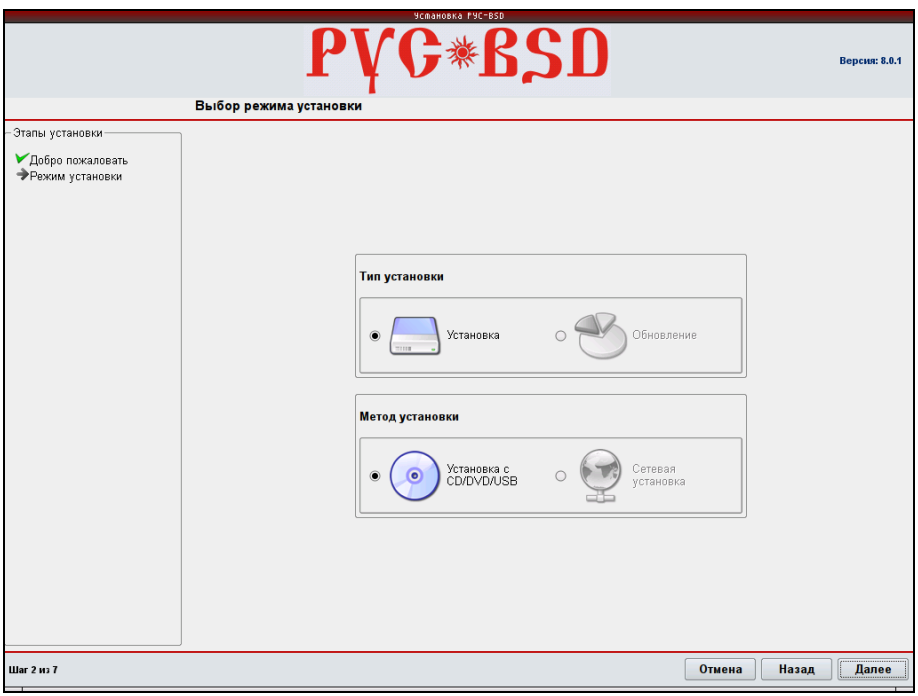


Рис. 4.5. Выбор типа установки

Глава 5



Резервное копирование. Быстрая переустановка BSD/Linux/Windows

5.1. Зачем нужно делать резервные копии?

К сожалению, даже самые новые компьютеры не совершенны. Они иногда ломаются. Причиной сбоя может быть все, что угодно, например, банальный перепад напряжения, из-за которого выходит из строя жесткий диск.

Делать резервные копии полезно не только на сервере, но и на обычной рабочей станции или домашнем компьютере. Резервные копии помогут вам не только в случае полного уничтожения данных. Представьте, что вы нечаянно удалили какой-то важный файл или изменили его не так, как нужно, а после этого выполнили команду **Файл | Сохранить**. В первом случае (удаление) файл еще можно восстановить, но только если вы сразу обнаружили пропажу файла. Спустя некоторое время, скажем, через неделю, восстановить удастся лишь часть файла (или вообще ничего), поскольку блоки, в которых он размещался, уже физически перезаписаны другими данными. Частичное восстановление имеет смысл разве что в случае с простыми текстовыми файлами. Однако все современные текстовые процессоры хранят данные не в текстовом формате, а в двоичном, что связано в первую очередь с внедрением в файл двоичных данных (тех же рисунков). Поэтому восстановление части такого файла ничего вам не даст, и файл окажется потерянным навсегда. А при наличии резервной копии восстановить файл не составляет большого труда — вы просто перепишете его оттуда. Не правда ли, хороший выход из положения?

Резервная копия, хранящая исходный документ, поможет вам и в том случае, когда вы изменили файл и сохранили изменения (как правило, после сохранения изменений в большинстве приложений функция отмены последнего действия не работает). Можно, конечно, заново все переделать (скажем, напечатать повторно несколько страниц), но намного быстрее и удобнее восстановить оригинальный файл из резервной копии.

5.2. Выбор носителя для резервной копии

Делать резервную копию в другом разделе жесткого диска совершенно нет смысла, поскольку в случае его отказа вы не сможете прочитать не только свои данные, но и резервную копию. Резервные копии нужно хранить только на съемных носителях.

В свое время для создания резервных копий использовались *стримеры* — устройства, записывающие данные на магнитную ленту. Конечно, не на обычную магнитофонную — в стример устанавливалась специальная кассета с высококачественной магнитной лентой. Преимущество такого решения — его дешевизна. На кассету стримера можно было записать 2–3 гигабайта информации, что для того времени было рекордным значением, а сами устройства были весьма распространены и надежны. Недостатков, впрочем, тоже хватало: процесс создания резервной копии из-за того, что стримеры были устройствами довольно медленными, мог длиться часами, кассеты с резервными копиями нужно было очень бережно хранить, поскольку они, как и обычные магнитофонные кассеты, имели свойство размагничиваться, поэтому со временем важную информацию приходилось перезаписывать — обновлять. Это уже не говоря о возможной диверсии — испортить весь архив резервных копий мог один небольшой магнит.

С появлением лазерных компакт-дисков все изменилось. Конечно, пока приводы CD-RW стоили довольно дорого, для создания резервных копий продолжали использоваться стримеры или обычные дискеты (в домашних условиях). Да, на дискету много не запишешь, но ведь и объемы данных были не такими, как сейчас. Скажем, в 1995 году на несколько дискет в сжатом виде (текстовая информация очень хорошо сжимается) можно было записать практически все документы небольшой фирмы.

Спустя несколько лет приводы CD-RW, как и сами диски ("болванки"), существенно подешевели и стали доступны обычным пользователям. На CD можно было записать до 700 Мбайт информации в несжатом виде виде и до 1 Гбайт в сжатом (в зависимости от архиватора и от сжимаемой информации). Домашние пользователи, которые вообще раньше не делали резервных копий, начали активно использовать CD, хотя некоторые организации продолжали пользоваться стримерами — им так было проще.

Вымерли как вид стримеры только с появлением и удешевлением DVD. Преимущества DVD очевидны:

- на обычный DVD можно записать до 4,5 Гбайт информации, на двухслойный и двусторонний — до 17 Гбайт;
- скорость записи и чтения DVD значительно больше, чем стримера;
- DVD намного надежнее кассет стримера.

Но время шло, и объемы резервируемых данных нарастали лавинообразно. Представим, например, что у нас в ведении сервер с огромным жестким диском (или массивом жестких дисков) и базой данных Oracle. Размер базы данных — не много и не мало — 120 Гбайт. Записать 120 Гбайт на болванки DVD — занятие не для слабонервных, даже если предварительно сжать всю информацию. Дело в том, что двухсторонние и двухслойные диски для целей архивации данных применять неудобно, поэтому нам придется взять обычные односторонние, а это означает, что таких дисков потребуется более двух десятков. Значит, нужно искать другой носитель информации.

И тут на арене опять появляются… стримеры. Но уже современные. Правда, цена их, мягко говоря, "кусается". Хороший стример стоит от 350 до 650 долларов. Но стример — это всего лишь записывающее устройство, к нему нужны кассеты.

А вот примерная стоимость кассет: 160 Гбайт — 40–45, 400 Гбайт — 53–57, 800 Гбайт — 150–160 долларов. Цены, сами понимаете, зависят и от производителя кассеты, и от магазина, в котором вы их покупаете. Посчитаем: стример за 350 долларов и две кассеты по 45. Получается 440 долларов за 320 Гбайт информации. Согласитесь, не очень выгодно. Гораздо выгоднее купить внешний противоударный жесткий диск, цена на который начинается от 60 долларов за 320 Гбайт (например, от Silicon Power).

Что это — скрытая реклама? Может быть, но устройства этой китайской (простите, тайваньской) фирмы отличаются низкой ценой и хорошим качеством — проверено собственным опытом и желанием сэкономить. А внешний жесткий диск производства Seagate емкостью 1,5 Тбайт со скоростью вращения 7200 об/мин обойдется всего-то в 120 долларов. Так что как альтернативу DVD и стримерам можно использовать внешние жесткие диски. В случае необходимости восстановить резервную копию с такого диска можно очень просто — даже не нужно выключать питание компьютера, поскольку диск подключается по USB. А как же диски с горячей заменой? Да, есть и такие. Но они применяются в основном в серьезных серверах с соответствующей ценой. Если есть желание и необходимость потратить значительную сумму — покупайте и используйте диски с горячей заменой. Мне же проще и дешевле использовать внешний жесткий диск.

Итак, можно резюмировать, что для резервного копирования небольших объемов данных (до 4 Гбайт) вполне подойдут DVD или DVD-RW. Можно использовать и другие съемные носители, которые есть под рукой: Flash-диски, магнитооптические носители. Если же требуется резервировать десятки, а то и сотни гигабайт, поможет только внешний жесткий диск соответствующей емкости.

5.3. Правила хранения DVD с резервными копиями

Ваша резервная копия на DVD будет жить долго, если вы будете придерживаться следующих простых правил:

1. На носитель с резервной копией не нужно записывать посторонние данные.
   Предположим, что вы решили записать на DVD резервную копию своих документов общим объемом 1 Гбайт, и при этом оказалось, что на диске осталось свободными 3,5 Гбайт. Существует соблазн использовать свободное место по прямому назначению и дописать диск до конца — записать еще, например, фильм или музыку. Ни в коем случае не стоит этого делать. Рано или поздно вы захотите послушать музыку или посмотреть фильм или, возможно, одолжите диск приятелю, чтобы фильм посмотрел и он. В результате диск может быть утерян или поврежден. Помните, диском с резервной копией нужно пользоваться только тогда, когда эта копия вам необходима.
2. Не нужно дописывать на диск вторую резервную копию.
   Опять-таки на диске осталось еще много свободного места, и спустя некоторое время после записи первой резервной копии вы захотите дописать на него следующую. Не нужно делать и этого — чем меньше мы используем диск, тем

меньше он изнашивается, следовательно, тем лучше (см. правило 1). Впрочем, из этого правила есть исключения, диктуемые здравым смыслом и стратегией копирования. Об этом мы еще поговорим.

3. Никогда не доверяйте диски с важными данными посторонним людям.
   Во-первых, это не желательно с точки зрения конфиденциальности данных, во-вторых, важные резервные копии могут быть просто утеряны.
4. Храните диски в темном сухом помещении и обязательно каждый в отдельном боксе.
   На CD/DVD диски (как и на магнитооптические носители, не говоря уже о лентах стримеров) негативно влияют солнечные лучи. Поэтому диски с резервными копиями следует убрать подальше от прямого попадания солнечных лучей и взглядов посторонних. Магнитооптические носители, чтобы избежать размагничивания диска, держите подальше от источников магнитного излучения. Каждый диск надо хранить в отдельном боксе — не храните диски в круглых коробках, в которых диски нанизываются на шкив, размещенный по центру коробки.
5. Подписывайте ваши носители.
   Указывайте дату и время копирования, а также, по возможности, что находится на диске (это нужно писать на бумажной обложке бокса). Для надписи на поверхности диска можно использовать маркер. Все это облегчит поиск нужной резервной копии.

5.4. Стратегии создания резервной копии

Существует несколько стратегий создания резервных копий. Одна из них предполагает создание копии всего жесткого диска с последующей записью на отдельные носители только изменившихся данных. Спустя некоторое время (от недели до месяца — зависит от количества новых данных) опять делается копия всего жесткого диска и т. д. Такая стратегия идеально подходит для сервера предприятия, но не для обычного домашнего пользователя. К сожалению, универсальной стратегии нет, поэтому остается предложить собственную.

Начнем с первого положения — копирование всего винчестера (никого уже не удивишь винчестером в 250 Гбайт, сегодня это, пожалуй, минимум). Как уже отмечалось, для резервирования таких объемов DVD неприменимы. А вот делать ли полную копию винчестера на внешний жесткий диск, зависит от важности данных. Конечно, если все 250 Гбайт — важные, придется делать копию всего диска. Тем не менее, включать в резервную копию программные файлы смысла я не вижу — их легко восстановить с дистрибутивных дисков.

Итак, на носитель резервной копии следует записывать:

- конфигурационные файлы системы — просто каждый раз при создании резервной копии записывайте на диск каталоги /etc и /usr/local/etc;
- измененные пользовательские данные — если не помните, что вы изменяли (с какими документами работали), можно записать весь каталог /home (не забудьте о каталоге /root);

- каталог /usr/local/www/apache22/data — это корневой каталог Web-сервера, если, конечно, вы его используете. Как правило, Web-программисты тестируют на домашнем компьютере свои сценарии, поэтому копию данного каталога нужно сделать обязательно;
- рабочие каталоги других сетевых служб (например, FTP-сервера, если там есть что-то важное, или каталог с базой данных).

Прежде всего надо создать первую резервную копию. Как правило, она будет самая большая. А потом делать дополнительные резервные копии, в среднем — раз в неделю. Включать в них следует только изменившиеся данные. Если вы уверены, что не изменяли конфигурацию системы, можно не записывать каталог /etc (или /usr/loca/etc). А если вы просто работали с документами, зарезервируйте только свой домашний каталог.

Для такой схемы вам понадобятся два диска. Первый назовем *диском месяца* — он будет содержать полную копию (по приведенным пунктам). На диск недели вы будете еженедельно (возможно, чаще — все зависит от важности изменяемой информации) записывать изменившуюся информацию (в лучшем случае — просто каталог /home). На диск недели информацию нужно дописывать, то есть в конце месяца на этом диске будет минимум четыре каталога /home (названные, например, по номеру недели: home-1, home-2, home-3 и home-4). В начале следующего месяца на новый диск вы снова записываете полную копию, то есть создаете очередной диск месяца, и схема повторяется. Здесь все просто — думаю, не запутаетесь.

Лично я предпочитаю создавать резервные копии на основе ISO-образов LiveCD *(см. разд. 5.5)*. Сначала устанавливаю систему, настраиваю службы — здесь надо убедиться, что все работает без нареканий. Потом создаю образ диска с системой — это позволит мне быстро переустановить систему после сбоя (сбои бывают различные: вышел из строя жесткий диск, систему взломали — всякое случается...). Вне зависимости от ситуации я знаю, что имея образ диска с системой, за считанные минуты смогу восстановить систему. Для создания ISO-образа LiveCD (резервной копии системы) использую программу Clonezilla *(см. разд. 5.6)*. Потом для восстановления системы понадобится только диск с резервной копией и ничего больше. Если размер резервной копии огромен, можно поместить ее на другой жесткий диск (тогда для восстановления системы понадобится загрузочный диск Clonezilla). Кстати, программу Clonezilla можно использовать для резервирования не только FreeBSD, но и Linux, и Windows — в общем, это идеальный инструмент для любого администратора.

После создания системной резервной копии (диск месяца) можно приступить к созданию еженедельных резервных копий по схеме, описанной ранее. Для этого Clonezilla уже не нужна — создавать копию всей системы нам до следующего месяца не придется. Идеально подходит классическая программа tar *(см. разд. 5.7)*. Созданные программой tar копии можно записать на сменный носитель: DVD, внешний жесткий диск, кассету стримера.

5.5. "Живая" резервная копия

Уточним, зачем нам нужны средства для создания LiveCD. Ведь мы говорим о резервном копировании системы — причем здесь LiveCD? Оказывается, это довольно удобно. Мы убиваем вот столько зайцев сразу:

- Создаем средство для восстановления системы.

  Предположим, вы настроили свою систему, "подняли" все сетевые службы, отредактировали их конфигурационные файлы. Но завтра из-за очередного перепада напряжения сгорел жесткий диск. Опять все заново устанавливать/настраивать? Если вы накануне создали LiveCD, то вам нечего беспокоиться — заменили жесткий диск, загрузились с LiveCD (учитывая размер системы, это, скорее всего, будет уже LiveDVD, но по старинке мы здесь и далее будем называть такой диск LiveCD) и восстановили систему вместе со всеми параметрами на новый винчестер. На всю эту операцию будет потрачено полчаса, от силы минут сорок, и это вместе с установкой нового жесткого диска. Пользователи и начальство будут вам благодарны за столь оперативное "воскрешение" сервера. А теперь представьте, что вы создали обычный "бэкап" с помощью архиваторов tar/tgz. Вам понадобится минимум 40 минут на установку системы, потом придется потратить время на восстановление резервной копии плюс одна лишняя перезагрузка. Однозначно времени будет затрачено больше.

- Создаем средство для клонирования системы.

  Когда предприятие обзаводится компьютерным парком, то приобретаются, как правило, компьютеры однотипные. Исключение составляют разве что серверы — они должны быть мощнее, и компьютеры начальства — их следует оснастить мощной видеокартой ☺. И вот теперь представьте, что вам нужно последовательно настроить все новые компьютеры, а их может быть десять, двадцать, пятьдесят! Есть более простой путь: настроить один компьютер, создать LiveCD установленной на нем системы и развернуть ее на всех остальных компьютерах сети. Пусть отладка (установка системы и ее настройка) одного компьютера займет полтора часа, создание LiveCD — еще минут тридцать (тут все зависит от производительности компьютера, потому что от вас потребуется ввести всего одну команду), понадобится еще сколько-то времени для записи образа на несколько болванок. Да, именно на несколько, потому что придется создать несколько копий LiveCD, чтобы вы могли одновременно устанавливать систему на несколько компьютеров. Затем еще минут сорок ожидания, и будет отлажено N компьютеров сразу (число N зависит от количества записанных болванок). Удобно? Думаю, да. Без LiveCD вы бы затратили полтора часа на каждый компьютер. 10 компьютеров — это 15 часов или два рабочих дня. А так эта работа займет примерно 4 часа. Созданные "клоны" системы можно использовать и в будущем, если компьютерный парк станет расширяться.

- Получаем возможность создания LiveUSB.

  Средства создания LiveCD позволяют также создать и загрузочную флешку. Загрузочная "живая" флешка понадобится для восстановления/клонирования ОС нетбука и других компьютеров, не имеющих привода DVD.

Не нужно думать, что "бэкап" в виде LiveCD может использоваться только для копирования/восстановления файлов самой системы. Можно копировать и пользовательские данные из /home, лишь бы их размер не превысил размера DVD (впрочем, несмотря на некоторые неудобства, можно использовать двухслойные диски, что позволит увеличить объем резервируемой информации).

5.6. Клонирование дисков — программа Clonezilla

Многие знают "привидение от Нортона" — Symantec Norton Ghost. В мире Windows это продукт незаменимый. Здесь же мы познакомимся с программой Clonezilla — бесплатным (в отличие от продукта Symantec) аналогом Ghost.

Clonezilla является самым мощным средством для клонирования FreeBSD/Linux. Программа может не только создать LiveCD, но и развернуть систему по сети.

> ***Примечание***
>
> На сайте разработчиков **http://clonezilla.org/** приведена следующая информация: за 10 минут Clonezilla SE (SE, Server Editioin) развернула по сети образ объемом 5,6 Гбайт на 41 компьютер сети. То есть за эти 10 минут все компьютеры оказались подготовлены к работе. Правда, для такой сетевой установки нужно развернуть специальный сервер, и мы этот вариант рассматривать не будем.

Рассмотрим основные возможности Clonezilla:

- полностью бесплатна (распространяется по лицензии GPL);
- поддерживает файловые системы ext2, ext3, ext4, reiserfs, reiser4, xfs, jfs, FAT, NTFS, HFS (Mac OS), UFS (FreeBSD, NetBSD, OpenBSD), VMFS (VMware ESX), поэтому вы можете клонировать не только FreeBSD, но и MS Windows, Mac OS (Intel), Linux, NetBSD и OpenBSD;
- поддерживает LVM2 (LVM ver. 1 не поддерживает);
- поддерживает GRUB версий 1 и 2;
- поддерживает Multicast для массового клонирования по сети при условии, что компьютеры поддерживают PXE и Wake-on-LAN — только версия Clonezilla SE (Server Edition);
- может сохранить не только отдельно взятый раздел, но и весь жесткий диск со всеми разделами.

> ***Примечание***
>
> Clonezilla — программа весьма мощная, но здесь мы рассмотрим лишь один из примеров ее использования, а именно — создание LiveCD и восстановление системы с его помощью. Познакомиться с остальными возможностями программы можно в документации или на сайте разработчиков.

Итак, для создания/восстановления нужно выполнить следующие действия:

1. Скачайте с **http://clonezilla.org/download/sourceforge/** ISO-образ Clonezilla Live и разверните его на болванку компакт-диска — получится загрузочный диск Clonezilla Live;

2. Загрузитесь с диска Clonezilla Live (рис. 5.1) и выберите опцию **Clonezilla live**. Если возникнут проблемы (например, с видеокартой), можно через опцию **Other modes of Clonezilla live** выбрать другой режим загрузки Clonezilla. Вы увидите процесс загрузки Debian — тут все как обычно, нужно просто подождать (рис. 5.2).
3. В следующем окне (рис. 5.3) вам предложат выбрать язык — русского, к сожалению, пока не предвидится. В окне выбора раскладки клавиатуры (рис. 5.4) выберите вариант **Don't touch keymap**, поскольку раскладку изменять нам ни к чему.
4. В открывшемся окне выберите опцию **Start Clonezilla** (рис. 5.5), а затем режим **device-image** — создание файла образа раздела (рис. 5.6). Режим **device-device** используется для создания образа раздела, при этом сам образ будет помещен на другой раздел.
5. Далее нужно выбрать устройство, куда будет сохранен образ (или откуда он будет прочитан в случае восстановления системы по образу). Выберите **local_dev**, что означает локальное устройство (рис. 5.7). Также образ можно получить (или записать) по SSH, NFS (Network File System, а не Need For Speed!) и из сети MS Windows (опция **samba_server**).
6. Затем следует выбрать раздел, где будут храниться образы. Если вы создаете образ, то на этот раздел он будет сохранен, а если восстанавливаете образ, то Clonezilla будет искать его на этом разделе.
7. В следующем окне (рис. 5.8) нужно выбрать одну из предлагаемых опций:
   - **savedisk** — служит для сохранения всего диска;
   - **saveparts** — для сохранения одного или нескольких разделов диска;
   - **restoredisk** — для восстановления образа диска на локальный диск;
   - **restoreparts** — для восстановления образа раздела;
   - **recovery-iso-zip** — для создания "живого" диска восстановления.
8. Если вы собираетесь восстанавливать систему из образа, то в следующем окне (рис. 5.9) необходимо выбрать образ, который нужно для этого использовать.
9. А в следующем окне (рис. 5.10) — ввести устройство (имена устройств соответствуют именам устройств в Linux), на которое надо развернуть образ. Будьте внимательны, чтобы не развернуть образ раздела на весь диск — потеряете остальные разделы!
10. Если вы выбрали опцию **recovery-iso-zip** (рис. 5.11) для создания LiveDVD/USB, то нужно также выбрать режим:
    - **iso** — будет создан образ для записи на DVD;
    - **zip** — будет создан образ для записи на LiveUSB;
    - **both** — будут созданы оба файла образа, которые можно использовать впоследствии как для создания LiveDVD, так и для создания LiveUSB.

    Созданный файл (файлы) будет сохранен в каталоге /home/partimag (рис. 5.12).

На рис. 5.13 изображен процесс создания LiveCD, а из рис. 5.14 видно, что этот процесс удачно завершен.

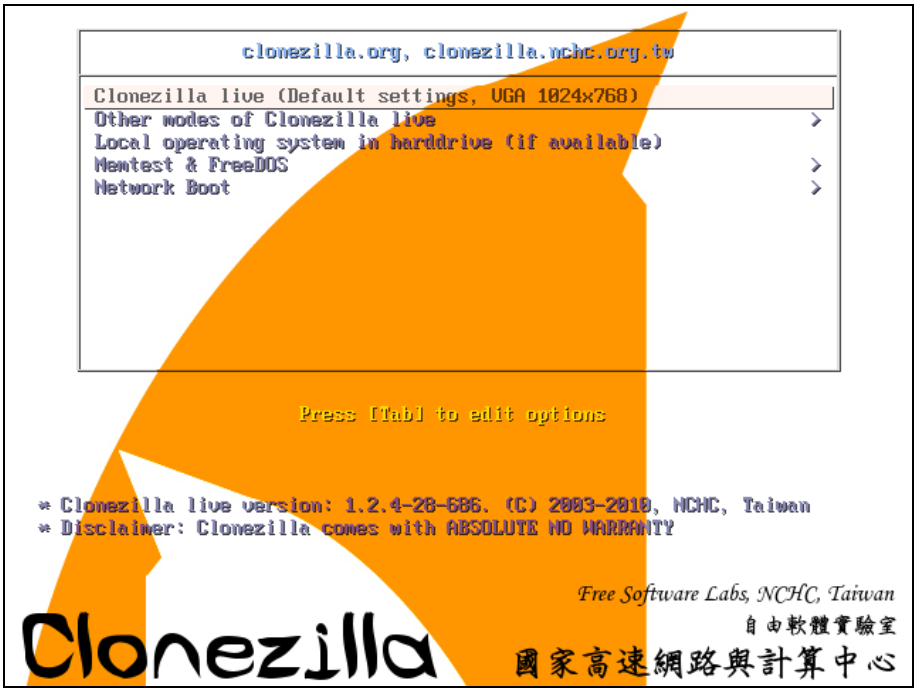


Рис. 5.1. Загрузочное меню Clonezilla Live

```
[    2.298874] scsi 1:0:1:0: Direct-Access     ATA      VMware Virtual I 0000 PQ: 0 ANSI: 5
[    2.340195] ata2.00: ATAPI: VMware Virtual IDE CDROM Drive, 00000001, max UDMA/33
[    2.341583] ata2.00: configured for UDMA/33
[    2.342129] scsi 2:0:0:0: CD-ROM            NECVMWar VMware IDE CDR10 1.00 PQ: 0 ANSI: 5
[    2.350556] sr0: scsi3-mmc drive: 1x/1x xa/form2 cdda tray
[    2.352065] Uniform CD-ROM driver Revision: 3.20
[    2.358812] sd 1:0:0:0: [sda] 16777216 512-byte logical blocks: (8.58 GB/8.00 GiB)
[    2.359612] sd 1:0:0:0: [sda] Write Protect is off
[    2.361466] sd 1:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or
FUA
[    2.362200]  sda: sda1 sda2 sda3 < sda5 >
[    2.363092] sd 1:0:1:0: [sdb] 31457280 512-byte logical blocks: (16.1 GB/15.0 GiB)
[    2.363185] sd 1:0:1:0: [sdb] Write Protect is off
[    2.363228] sd 1:0:1:0: [sdb] Write cache: disabled, read cache: enabled, doesn't support DPO or
FUA
[    2.380613]  sdb: sdb1
[    2.386360] sd 1:0:1:0: [sdb] Attached SCSI disk
[    2.387994] sd 1:0:0:0: [sda] Attached SCSI disk
[    2.391994] sd 1:0:0:0: Attached scsi generic sg0 type 0
[    2.393897] sd 1:0:1:0: Attached scsi generic sg1 type 0
[    2.400551] sr 2:0:0:0: Attached scsi generic sg2 type 5
Begin: Loading essential drivers ... [    2.593615] Atheros(R) L2 Ethernet Driver - version 2.2.3
[    2.593830] Copyright (c) 2007 Atheros Corporation.
[    2.612151] Broadcom NetXtreme II 5771x 10Gigabit Ethernet Driver bnx2x 1.52.1 (2009/08/12)
[    2.632202] device-mapper: uevent: version 1.0.3
[    2.634009] device-mapper: ioctl: 4.15.0-ioctl (2009-04-01) initialised: dm-devel@redhat.com
done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... [    2.745155] Uniform Multi-Platform E-IDE driver
[    2.745836] ide_generic: please use "probe_mask=0x3f" module parameter for probing all legacy ISA
 IDE ports
[    2.882403] aufs: module is from the staging directory, the quality is unknown, you have been war
ned.
[    2.885440] aufs 2-standalone.tree-32-20100125
[    2.930106] loop: module loaded
[    3.041203] squashfs: version 4.0 (2009/01/31) Phillip Lougher
```

Рис. 5.2. Процесс загрузки Debian

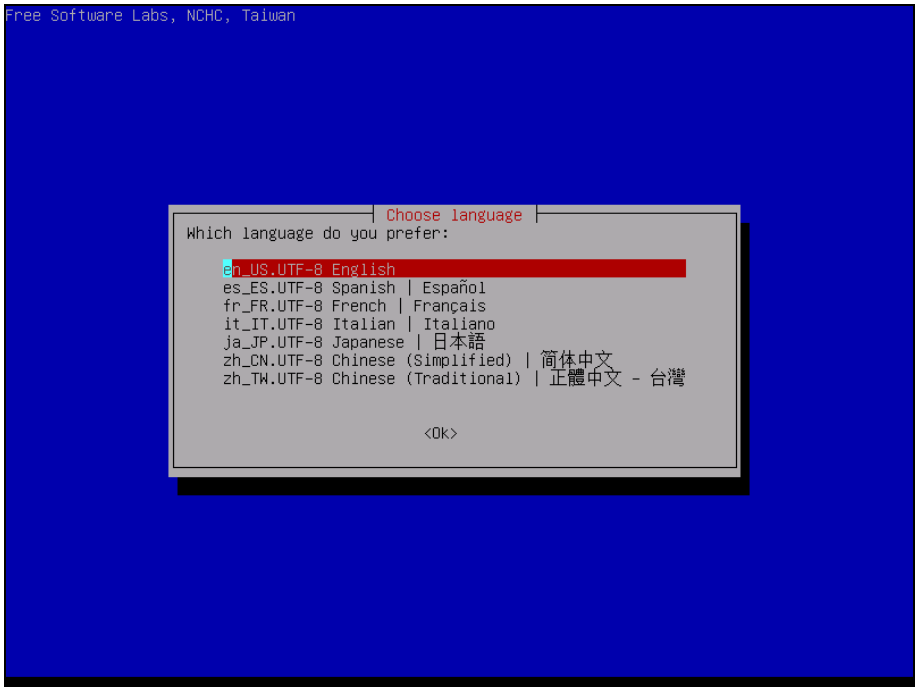


Рис. 5.3. Выбор языка Clonezilla

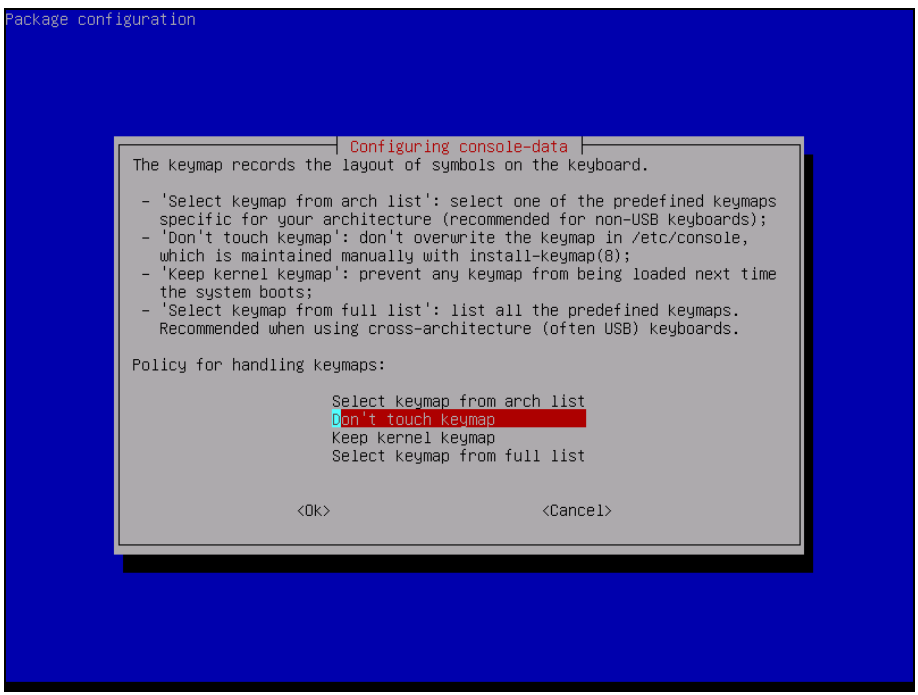


Рис. 5.4. Выбор раскладки

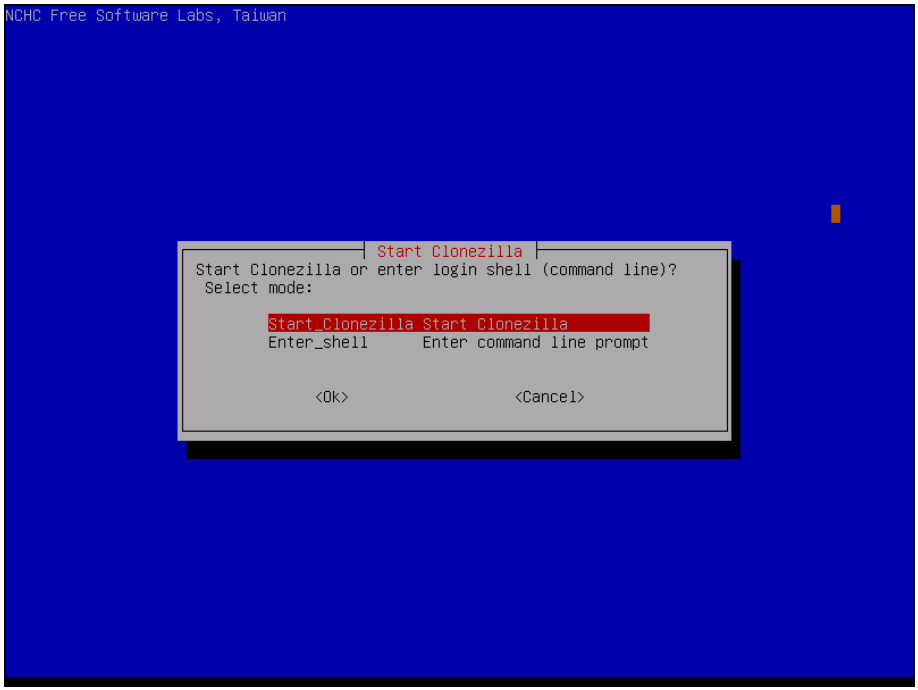


Рис. 5.5. Выберите опцию **Start Clonezilla**

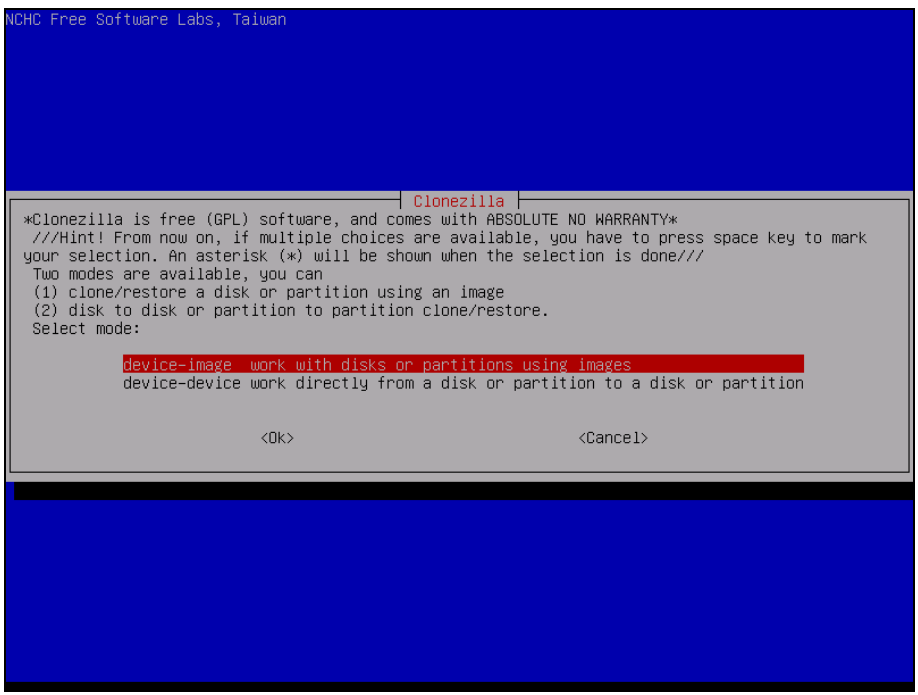


Рис. 5.6. Выберите режим **device-image**

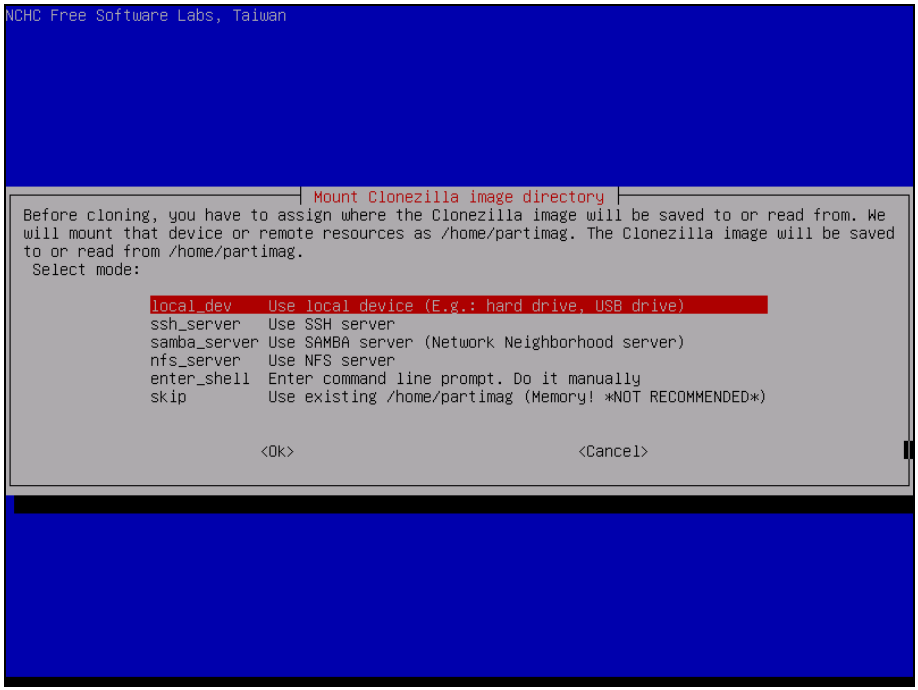


Рис. 5.7. Выбор носителя образа

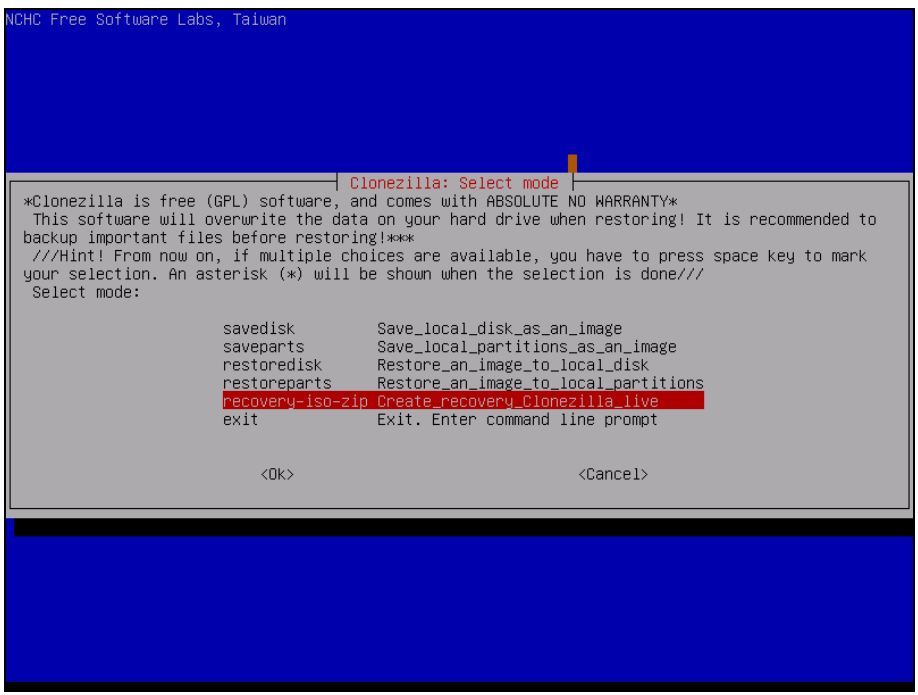


Рис. 5.8. Создать образ или восстановить?

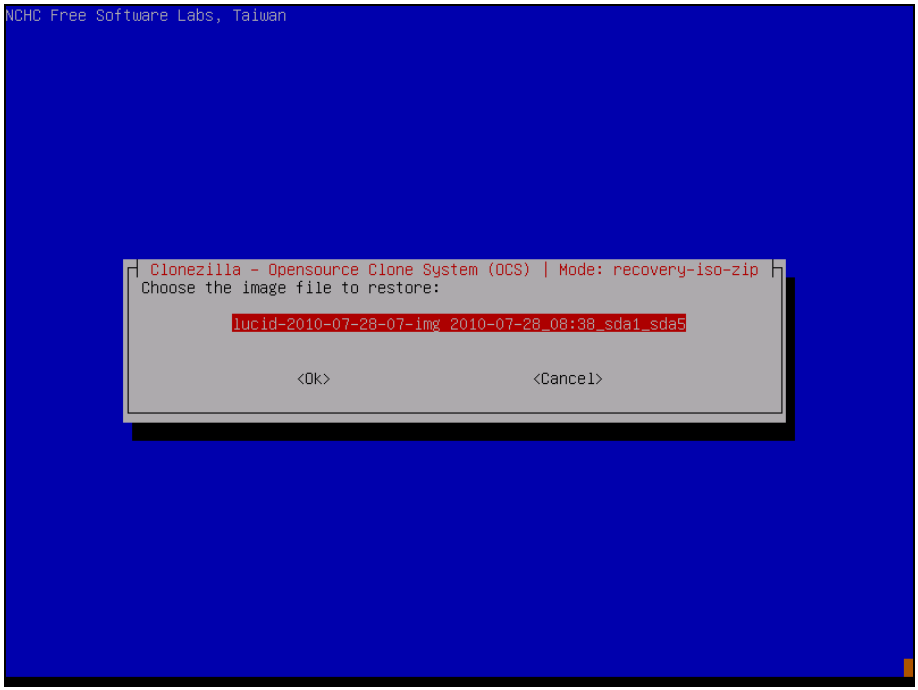


Рис. 5.9. Выбор образа для восстановления

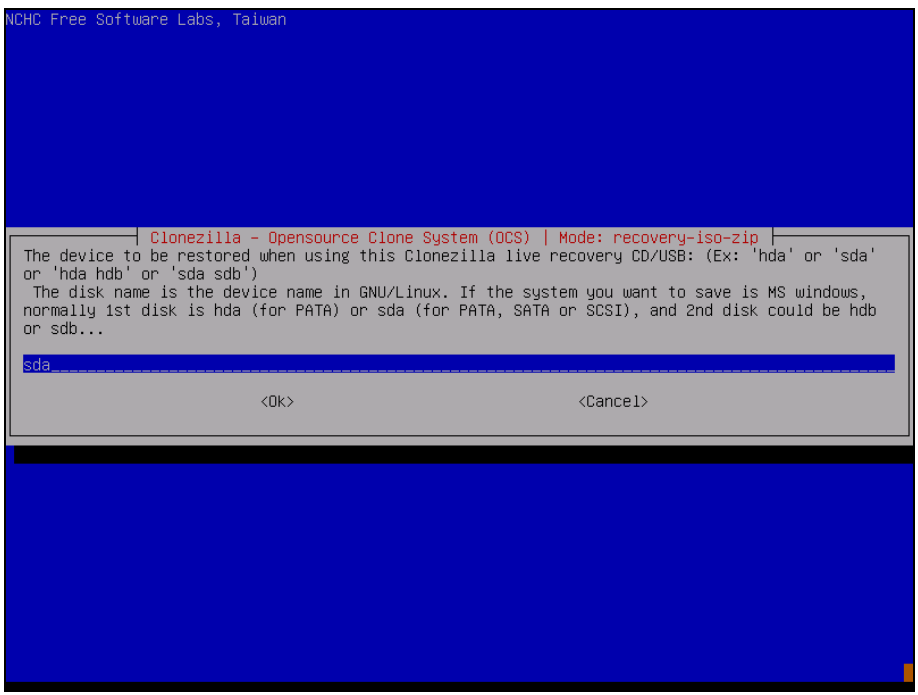


Рис. 5.10. На какое устройство развернуть образ

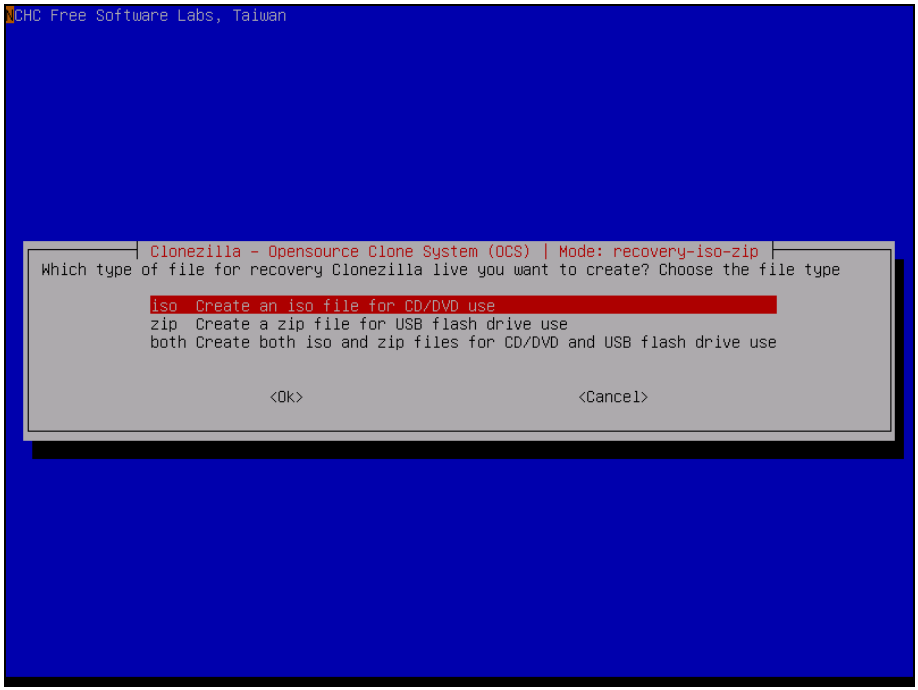


Рис. 5.11. Выбор режима опции **recovery-iso-zip**

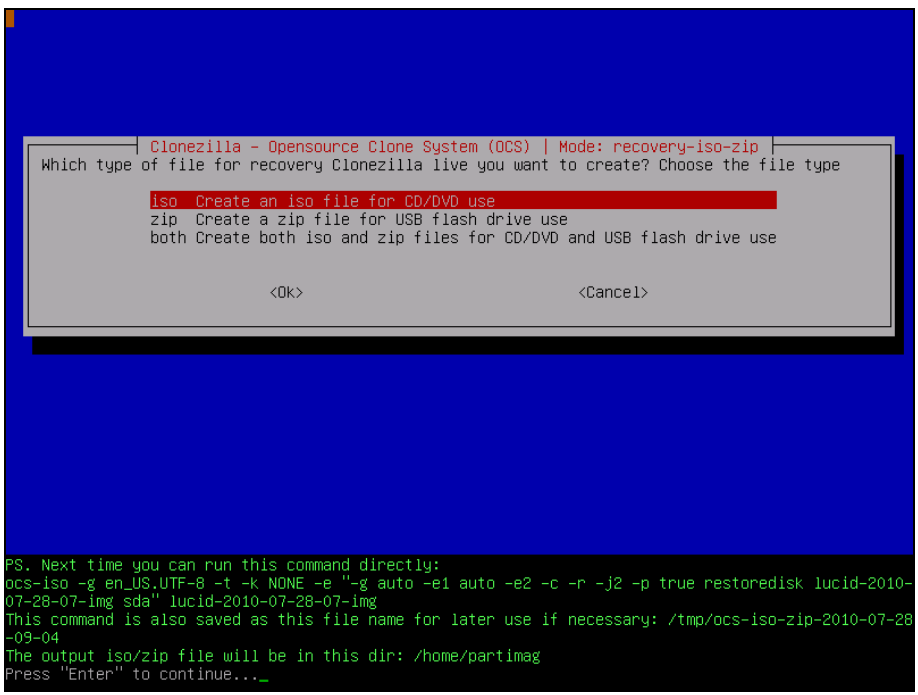


Рис. 5.12. Созданный файл будет сохранен в каталоге /home/partimag

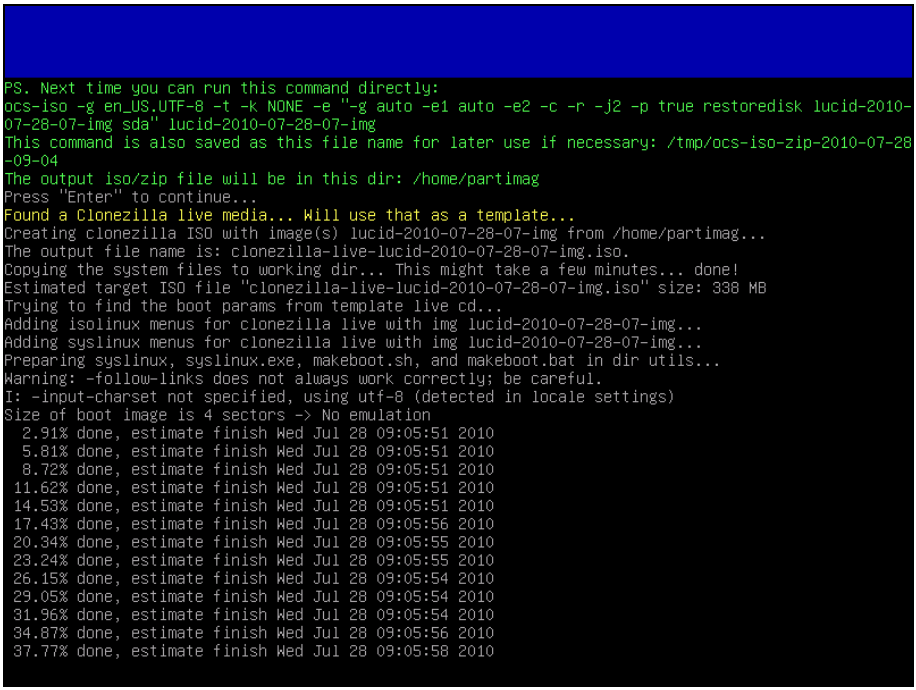


Рис. 5.13. Процесс создания LiveCD

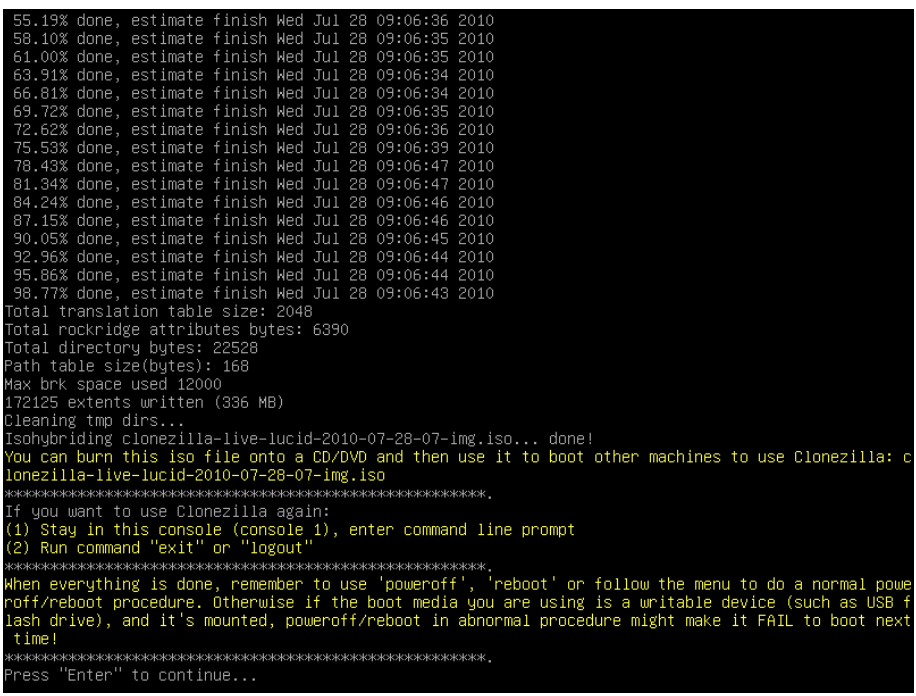


Рис. 5.14. LiveCD создан, нажмите клавишу <Enter> для продолжения

Вот и все! Как видите, все довольно просто. Программа работает с устройствами (дисками, разделами) напрямую, поэтому при создании/восстановлении образа все равно, под какой операционной системой работает компьютер.

Если у вас есть необходимость в серверной версии Clonezilla Server Edition, прочитать руководство по ее использованию вы можете по адресу: **http://clonezilla.org/clonezilla-server-edition/**.

5.7. Программа tar

Программу tar можно использовать для создания архива резервной копии — с одним архивом работать проще, чем с тысячей файлов, которые нужно поместить в резервную копию, да и место на DVD сэкономим.

Мы не будем изучать все опции tar — их достаточно много (о них вы можете прочитать в руководстве по команде `man tar`), а рассмотрим только команду, позволяющую заархивировать нужный нам каталог:

```
tar -cvjf имя_архива.tar.bz2 каталог
```

Например,

```
tar -cvjf homes.tar.bz2 /home
```

Рисунок 5.15 иллюстрирует процесс архивации.

Чтобы разархивировать архив, перейдите в каталог, в который вы хотите его распаковать, и введите команду:

```
tar -xvjf имя_архива.tar.bz2
```

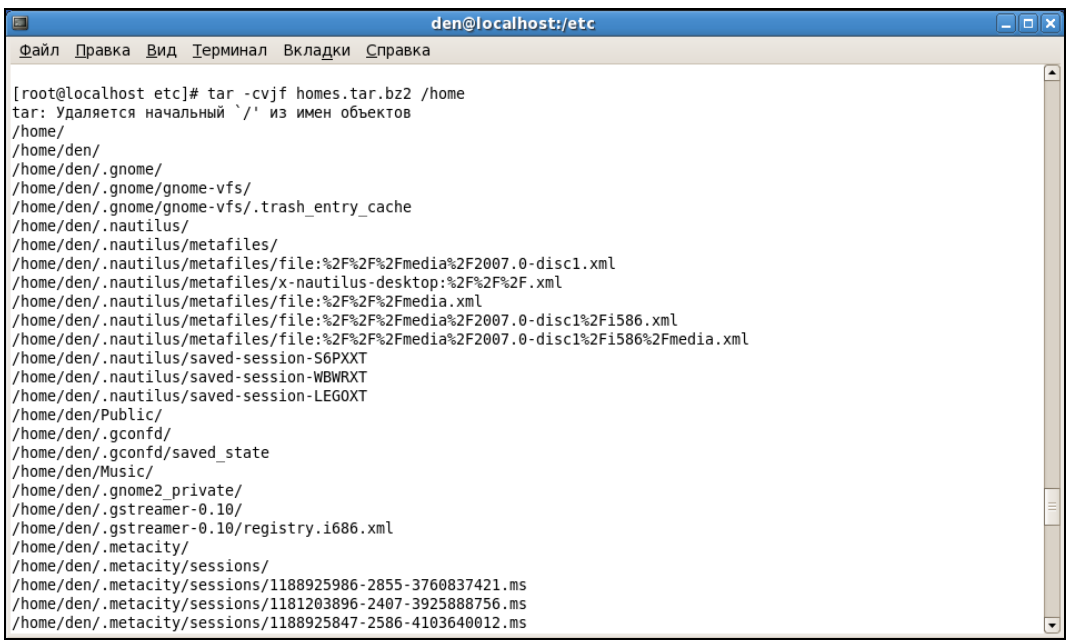


Рис. 5.15. Архивация

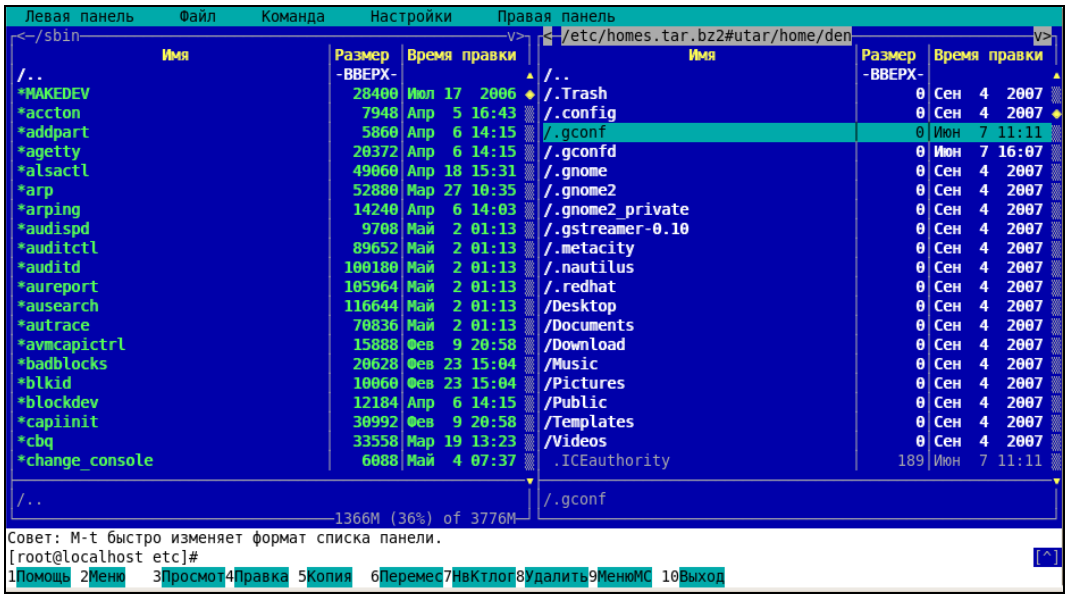


Рис. 5.16. Работа с архивом в mc

Если необходимо извлечь всего пару файлов, проще использовать файловый менеджер mc (рис. 5.16). Пакет тоже называется mc, и его легко можно установить в систему *(см. главу 18)*.

5.8. Сетевое резервное копирование

Задача проста — в вашей сети есть компьютеры, на которых развернуты, например, Web-сервер и почтовый сервер. Нужно сделать резервные копии данных, хранимых на этих компьютерах. Понятно, что системному администратору отлучаться от своего рабочего места желания нет. Поэтому вы можете создать резервную копию по сети с помощью команды `scp`:

```
scp -r имя_каталога компьютер:каталог
```

Например,

```
scp -r web-cp web-server:/var/www
```

Команда `scp` (secure copy) используется для безопасного копирования файлов по сети. Для того чтобы она работала, на удаленном компьютере должен быть установлен сервис sshd.

Вернемся к нашей команде. Параметр `-r` означает, что нужно скопировать подкаталоги удаленного каталога, то есть осуществить рекурсивное копирование. После него задается имя локального каталога (`web-cp`), куда будут записаны скопированные файлы и каталоги. `web-sever` — это имя удаленного компьютера (можно задать IP-адрес), а через двоеточие указан удаленный каталог, который вы хотите скопировать.

Вам осталось лишь заархивировать каталог web-cp:

```
tar -cvjf web-cp.tar.bz2 web-cp
```

5.9. Запись DVD-болванок в FreeBSD

Запись болванок в FreeBSD проблем не вызывает. Сам процесс записи подобен аналогичному процессу в Linux. Но отличия все же есть. Первым делом нам нужно включить прямой доступ для устройств ATAPI, коими являются DVD-приводы. Для этого в файле /boot/loader.conf укажите:

```
hw.ata.atapi_dma="1"
```

Далее нужно установить пакет sysutils/dvd+rw-tools *(см. главу 18)*. В этом пакете имеется утилита `growisofs`, которая используется для записи болванок. Рассмотрим несколько примеров использования этой утилиты. Например, вам нужно записать каталог /home/denis на устройство /dev/cd0 (первый DVD-привод):

```
# growisofs -dvd-compat -Z /dev/cd0 -J -R /home/denis
```

> ***Совет***
>
> Чтобы разобраться в опциях команды `growisofs`, обратитесь к справочной системе (`man growisofs`).

Программа `growisofs` может также записать уже готовый ISO-образ (мы запишем образ image.iso):

```
# growisofs -dvd-compat -Z /dev/cd0=image.iso
```

Перед записью DVD-RW/DVD+RW носитель желательно отформатировать. Делается это командой:

```
# dvd+rw-format /dev/cd0
```

Мы отформатировали диск так, что дозапись на него уже невозможна. Если вы желаете создать диск с возможностью дозаписи, используйте другую команду:

```
# dvd+rw-format -blank=full /dev/cd0
```

Очистить (не отформатировать, а просто очистить!) диск можно вот такой командой:

```
# growisofs -Z /dev/cd0=/dev/zero
```

При первой записи диск нужно отформатировать, при последующих — можно просто очищать. Если с диском возникли проблемы, попробуйте снова его отформатировать.

Глава 6



Создание LiveCD своими руками

6.1. Создание дампа. Команда *dump*

В *главе 5* мы создали LiveCD с резервной копией системы средствами Clonezilla. В этой главе мы создадим то же самое, но средствами самой операционной системы. Сразу скажу: эта глава ориентирована на фанатов BSD и на тех пользователей, кто желает более глубоко освоить систему. Если вас интересует результат и оптимизация времени — используйте Clonezilla, это будет быстрее и проще. Для создания LiveCD вручную будет потрачено больше времени, зато вы узнаете много полезных команд и научитесь создавать дампы системы.

Прежде чем приступить к созданию LiveCD, разберемся, как будет работать наша самодеятельная система восстановления. Clonezilla позволяет создавать дамп (образ) жесткого диска как на другом диске, так и непосредственно на DVD. Если жесткий диск большой, то как бы мы его ни сжимали, вряд ли получится записать дампы (даже в сжатом виде) на DVD. Придется копировать их на другой жесткий диск, который в случае восстановления системы будет подключен к нашему компьютеру. Тогда мы загружаемся с диска Clonezilla, выбираем раздел диска, где содержатся дампы нашей системы, и восстанавливаем ее. Именно такую схему мы сейчас и реализуем: создадим дампы нашей системы, поместим их на другой жесткий диск, затем создадим LiveCD с FreeBSD. Для восстановления системы нужно будет загрузиться с этого диска, подмонтировать жесткий диск, содержащий дампы системы, и развернуть дампы. После перезагрузки система будет восстановлена.

Итак, первым делом создадим дампы системы. Как правило, наш BSD-слайс содержит следующие BSD-разделы:

/dev/da0s1a /

/dev/da0s1d /usr

/dev/da0s1e /var

BSD-разделы, используемые для подкачки и каталога /tmp, помещать на LiveCD не нужно, поэтому в нашем списке их нет.

Для создания дампов разделов нужно ввести команды:

```
# /sbin/dump -0ua -L -f- /dev/da0s1a | bzip2> /root.img.bz2
# /sbin/dump -0ua -L -f- /dev/da0s1d | bzip2> /usr.img.bz2
# /sbin/dump -0ua -L -f- /dev/da0s1e | bzip2> /var.img.bz2
```

Команда `dump` подготовит дампы разделов, которые потом будут сжаты архиватором `bzip2`. После создания дампов нужно переместить их на другой носитель. Для этого подмонтируем другой раздел — на другом жестком диске (в данном случае это раздел /backup на устройстве /dev/da1s1a):

```
# mkdir /backup
# mount /dev/da1s1a /backup
# mv /root.img.bz2 /backup
# mv /usr.img.bz2 /backup
# mv /var.img.bz2 /backup
# umount /backup
```

Можно создать сценарий, автоматизирующий создание дампов и перемещение их на другой жесткий диск. В *главе 12* вы научитесь создавать сценарии оболочки.

6.2. Создание LiveCD. Утилита FreeSBIE

Пора приступить ко второму этапу нашей задачи. Для создания LiveCD мы будем использовать порт FreeSBIE, но перед установкой этого порта нам нужно установить порты sysutils/dvd+rw-tools и sysutils/cdrtools:

```
# cd /usr/ports/sysutils
# cd cdrtools
# make install clean
# cd ..
# cd dvd+rw-tools
# make install clean
```

Теперь можно установить утилиту для создания LiveCD:

```
# cd /usr/ports/sysutils/freesbie
# make install clean
```

Далее введите команду:

```
# cp  /usr/local/share/freesbie /usr/local/freesbie
```

Для создания LiveCD введите команды:

```
# cd /usr/local/freesbie
# make iso
```

Начнется создание ISO-образа диска:

Building world for i386 architecture
>>> Rebuilding the temporary build tree
>>> stage 1.1: legacy release compatibility shims
>>> stage 1.2: bootstrap tools
>>> stage 2.1: cleaning up the object tree
...
ISO created:
-rw-r--r-- 1 root wheel   86M 16 окт 11:34 /usr/obj/FreeSBIE.iso

Чтобы получить LiveCD минимального размера (чего вполне хватит для восстановления системы), откройте файл /usr/local/freesbie/conf/freesbie.defaults.conf и добавьте в него строку:

```
MINIMAL=YES
```

Понятно, что это нужно сделать до сборки ISO-образа.

6.3. Восстановление системы. Команда *restore*

Итак, у нас есть созданные дампы системы (на устройстве da1s1a) и загрузочный диск. Для восстановления системы нужно подключить жесткий диск, содержащий резервные копии (дампы) системы, загрузиться с LiveCD и начать вводить требуемые команды.

Сначала подмонтируем жесткий диск, содержащий дампы:

```
# mkdir /backup
# mount /dev/da1s1a /backup
```

Теперь подмонтируем жесткий диск с системой, которую нужно восстановить:

```
# mkdir /mnt/root
# mkdir /mnt/usr
# mkdir /mnt/var
# mount /dev/da0s1a /mnt/root
# mount /dev/da0s1d /mnt/usr
# mount /dev/da0s1e /mnt/var
```

Осталось только распаковать образы в каталоги /mnt/root, /mnt/usr и /mnt/var:

```
# cd /mnt/root
# bunzip2 --stdout /backup/root.img.bz2 | restore -rf -
# cd /mnt/usr
# bunzip2 --stdout /backup/usr.img.bz2 | restore -rf -
# cd /mnt/var
# bunzip2 --stdout /backup/var.img.bz2 | restore -rf -
```

Здесь образ сначала распаковывается программой `bunzip2`, а потом передается программе `restore`, которая и восстанавливает созданный образ. Как вы уже догадались, программа `restore` выполняет действия, обратные действиям программы `dump`.

Вот и все, осталось размонтировать устройства и перезагрузиться:

```
# cd /
# umount /backup
# umount /mnt/root
# umount /mnt/usr
# umount /mnt/var
# reboot
```

После перезагрузки ваша система будет восстановлена (только не забудьте извлечь DVD).



Глава 7

Особые варианты установки FreeBSD

7.1. Использование LiveUSB. Установка FreeBSD на нетбук

Этот раздел посвящен фанатам FreeBSD, у которых эта операционная система установлена на каждом находящемся в их распоряжении компьютере. Понятно, что любой уважающий себя гуру BSD захочет установить FreeBSD и на нетбук — мини-ноутбук без привода DVD. Кстати, приведенные здесь рекомендации можно использовать и для установки FreeBSD на сервер, не оснащенный приводом CD/DVD, — иногда на приводе экономят, а вот USB есть всегда. Главное, чтобы BIOS поддерживала загрузку с флешки, иначе ничего не получится.

В Интернете можно найти целые HOWTO по созданию загрузочной флешки и правильной установке FreeBSD на нетбук. Но, начиная с версии 8.0, эта информация утратила актуальность, поскольку теперь FreeBSD распространяется и в виде установочного образа для USB-флешки. Все, что потребуется — это скачать из FTP-каталога **ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/ISO-IMAGES/8.1/** файл FreeBSD-8.1-RELEASE-i386-memstick.img. Не забудьте только выбрать соответствующую архитектуру.

Далее нужно записать загруженный файл на флешку:

```
# dd if=8.1-RELEASE-i386-memstick.img of=/dev/da0 bs=10240 conv=sync
```

Здесь параметром `if` вы задаете загруженный файл, а параметром `of` — имя устройства флешки (будьте осторожны, чтобы нечаянно не перезаписать содержимое жесткого диска). Если у вас SCSI-диск, то ему будет присвоено имя /dev/da0, а флешке — /dev/da1. Чтобы узнать, какое имя присвоено вашей флешке, просмотрите файл /var/log/messages сразу после ее подключения:

```
# tail /var/log/messages
```

Вот пример сообщений в файле messages сразу после подключения USB-диска:

Oct 14 11:29:11 denhost kernel: ugen1.2: <Kingston> at usbus1

Oct 14 11:29:11 denhost kernel: umass0: <Kingston DataTraveler 2.0, class 0/0, rev 2.00/2.00, addr 2> on usbus1

Oct 14 11:29:11 denhost kernel: umass0: SCSI over Bulk-Only; quirks = 0x0000

Oct 14 11:29:11 denhost root: Unknown USB device: vendor 0x0951 product 0x1603 bus uhub1

Oct 14 11:29:12 denhost kernel: umass0:1:0:-1: Attached to scbus1

Oct 14 11:29:12 denhost kernel: uhub_explore:611: illegal enable change, port 1

Oct 14 11:29:12 denhost kernel: *da0* at umass-sim0 bus 0 target 0 lun 0

Oct 14 11:29:12 denhost kernel: *da0*: <Kingston DataTraveler 2.0 1.00> Removable Direct Access SCSI-2 device

Из этой записи видно, что нашей флешке присвоено имя /dev/da0.

После записи образа на флешку просто загрузитесь с нее. Проблем с загрузкой не будет, поскольку все нетбуки поддерживают загрузку с флешки. А дальше начнется обычный процесс установки, описанный в *главе 2*. В общем, остальное — дело техники.

7.2. Обновление до FreeBSD 8

Обладателям старых версий FreeBSD, скорее всего, захочется обновить их до версии 8.0. Сразу хочу предупредить, что это довольно долгий процесс. Намного проще инсталлировать FreeBSD 8 и заново ее настроить — установить все необходимые демоны и исправить конфигурационные файлы.

> ***Совет***
>
> Предварительно скопируйте все конфигурационные файлы на флешку или другой слайс жесткого диска. В любом случае обязательно сделайте резервную копию системы с помощью Clonezilla *(см. главу 5)* — даже если что-то пойдет не так, вы сможете легко восстановить все, как было.

Для обновления FreeBSD до версии 8.0-RELEASE (версию, понятно, можно изменить) служат команды (только в таком порядке):

```
# freebsd-update upgrade -r 8.0-RELEASE
# ee /etc/rc.conf
# freebsd-update install
# reboot
# freebsd-update install
# portupgrade -af -O
# freebsd-update install
# reboot
```

Во время процесса обновления придется два раза перезагрузить машину.

Перед обновлением настоятельно рекомендую внимательно прочитать следующий документ: **http://www.freebsd.org/releases/8.0R/relnotes-detailed.html**. В нем описаны возможные изменения в названиях устройств. Например, драйвер для сетевой карты, установленной на материнской плате NVIDIA nForce, раньше назывался nfe, а после обновления будет называться nve. Следовательно, если вы вовремя не измените файл /etc/rc.conf, то потеряете свой сервер. Не зря я после команды загрузки обновлений (загрузки, а не установки!) поместил команду редактирования файла rc.conf. Так вы сможете остановить (временно, потом "подниме-

те") все "лишние" демоны (Apache, FTP и т. п.) и внести изменения в описания сетевых интерфейсов. Так, в примере с NVIDIA nForce нужно заменить строку:

```
ifconfig_nve0="параметры"
```

строкой

```
ifconfig_nfe0="параметры"
```

В Интернете приводится множество примеров обновления FreeBSD разных версий. Процесс обновления у каждого пользователя будет свой, поскольку у каждого — свое "железо". Один из примеров — обновление версии 6.2 до версии 8.0 — вы можете найти по ссылке: **http://m.habrahabr.ru/post/77685/.**

7.3. Установка по сети

В *разд. 7.1* мы разобрались, как установить FreeBSD на любой компьютер без привода CD/DVD, — для этого использовался загрузочный образ флешки. Но не каждый компьютер может загружаться с флешки — не очень современные устройства не поддерживают загрузку через USB. Некоторые предприятия экономят — закупают серверы без приводов CD/DVD. Поэтому нужно искать альтернативный способ установки. Здесь мы рассмотрим установку FreeBSD по сети.

Для установки по сети нам понадобится еще один компьютер, работающий под управлением FreeBSD, — он будет выступать в роли загрузочного сервера. На нем мы установим серверы DHCP, TFTP, NFS. Обязательными условиями являются наличие установочного диска с FreeBSD и поддержка загрузки по сети сетевой картой компьютера, на который мы будем устанавливать FreeBSD. Перед установкой подумайте, может быть, проще временно доукомплектовать этот компьютер приводом CD/DVD? Правда, наличие пломб может несколько усложнить эту задачу, но раз есть пломбы, то компьютер — новый, следовательно, он поддерживает загрузку с флешки. Проверьте это.

Если же вы не ищете простых путей (правильно, на сложном пути всегда меньше конкурентов), тогда приступим к настройке загрузочного сервера. Первым делом установим и настроим DHCP-сервер. Введите команды:

```
# cd /usr/ports/net/isc-dhcp30-server
# make install clean
```

После этого откройте файл /usr/local/etc/dhcpd.conf и отредактируйте его так, как показано в листинге 7.1.

Листинг 7.1. Файл /usr/local/etc/dhcpd.conf

```
authoritative;
          subnet 192.168.0.0 netmask 255.255.255.0 {
          }
          host instsrv {
            # укажите MAC-адрес компьютера, на который
            # вы устанавливаете FreeBSD (клиента)
```

```
        hardware ethernet XX:XX:XX:XX:XX:XX;
        # IP-адрес клиента
        fixed-address 192.168.1.100;
        # IP-адрес этого сервера
        next-server 192.168.1.5;
        # Путь к загрузчику
        filename "boot/pxeboot";
        # Корневой каталог TFTP сервера
        option root-path "/tftpboot";
    }
```

Теперь "поднимем" сервер TFTP. Он уже установлен, поэтому нам нужно только раскомментировать следующую строку в файле /etc/inetd.conf:

```
tftp dgram udp wait root /usr/libexec/tftpd tftpd -l -s /tftpboot
```

Следующий шаг — настройка NFS. Для этого откройте файл /etc/exports и добавьте в него строку:

```
/tftpboot -network 192.168.0 -mask 255.255.255.0
```

Скопируйте содержимое установочного диска FreeBSD в каталог tftpboot (следующая команда предполагает, что диск подмонтирован к каталогу /mnt/cdrom):

```
# cp -Rp /mnt/cdrom /tftpboot
```

В файл /tftpboot/boot/loader.conf добавьте строку:

```
vfs.root.mountfrom="ufs:/dev/md0c"
```

Почти все. Осталось только отредактировать наш файл /etc/rc.conf, чтобы обеспечить запуск настроенных серверов и перезагрузиться. Добавьте в /etc/rc.conf строки:

```
# IP-адрес нашего сервера
ifconfig_em0="inet 192.168.0.5 netmask 255.255.255.0"
# Автоматический запуск DHCP
dhcpd_enable="YES"
# Автоматический запуск NFS и TFTP
rpcbind_enable="YES"
nfs_server_enable="YES"
inetd_enable="YES"
```

Перезагружаемся:

```
# reboot
```

Затем включаем компьютер, на который мы будем устанавливать FreeBSD. Когда начнется загрузка и будет предложено выбрать источник установки, выберите **NFS**, после чего введите расположение дистрибутива FreeBSD: `192.168.0.5:/tftpboot`.



ЧАСТЬ II

Настольное применение BSD



Глава 8

Настройка консоли

8.1. Вход в систему

Работа с системой начинается со входа в нее. После загрузки системы вы увидите приглашение войти в систему:

login:

Введите логин пользователя, например, `root`. Затем — пароль (рис. 8.1):

Password:

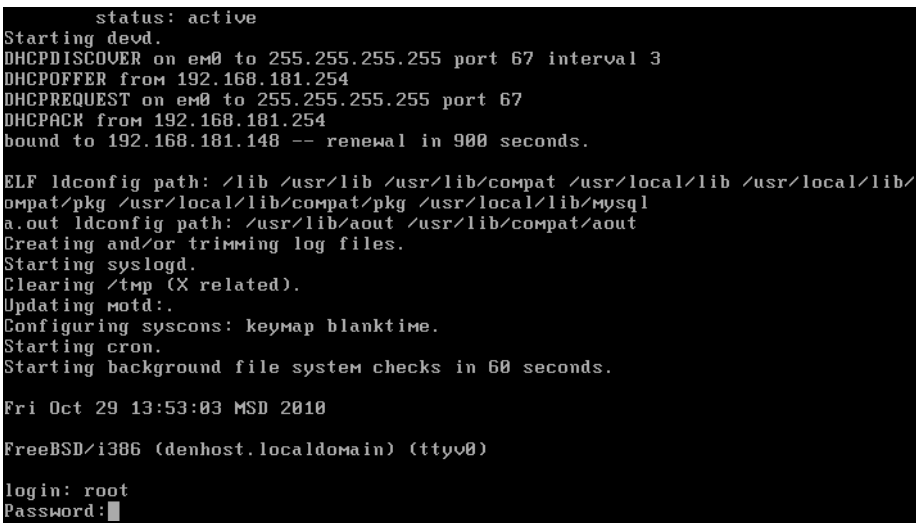


Рис. 8.1. Вводим имя пользователя и пароль

Если логин и пароль введены правильно, вы увидите приветственное сообщение (рис. 8.2) — так называемое *сообщение дня* (message of the day). Текст этого сообщения находится в файле /etc/motd, там его и можно изменить.

Ниже приветствия находится приглашение командного интерпретатора, обычно имеющее вид **имя_узла#**, — в эту строку вы можете вводить команды.

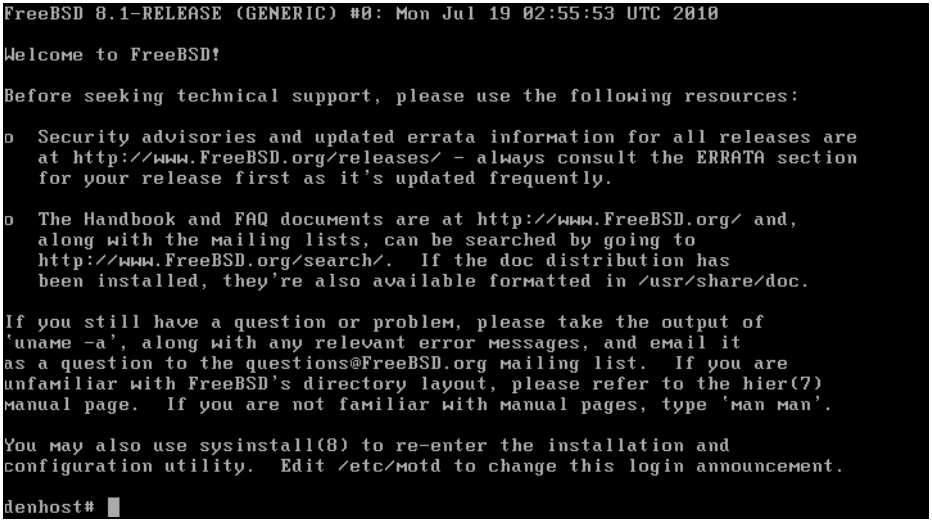


Рис. 8.2. Приветствие

8.2. Понятие о работе в консоли

Работа в консоли заключается во вводе нужной команды. Вы вводите команду (например, создания каталога, просмотра файла, вызова редактора и т. п.) и нажимаете клавишу <Enter>. Команда содержит как минимум имя запускаемой программы. Кроме имени программы команда может содержать параметры, которые будут переданы программе, а также символы перенаправления ввода/вывода *(см. разд. 11.5)*. Естественно, вам нужно знать имя программы, а также параметры, которые требуется ей передать.

Какие существуют команды, вы узнаете из этой книги. В каждой главе вы будете знакомиться с новыми для вас командами FreeBSD, особенно много полезных команд рассмотрено в *главе 13*. Пока же для тренировки введите команду `df -h`, отображающую информацию об использовании дискового пространства (рис. 8.3).

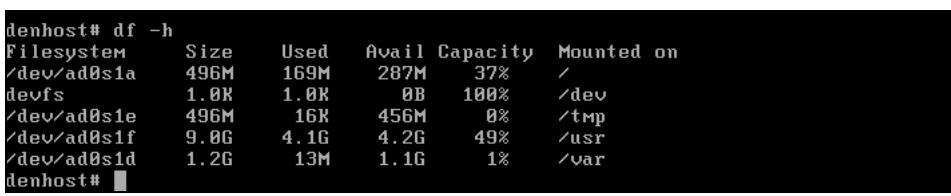


Рис. 8.3. Выполнение команды df -h

Если вы помните название программы, а назначение параметров забыли, поможет команда `man`. Man (от англ. manual) — это справочная система UNIX (FreeBSD, OpenBSD, Linux и других UNIX-подобных ОС). В ней имеется информация о каж-

дой установленной в системе программе. Откуда система знает обо всех программах? Все очень просто — разработчики программ договорились, что вместе с программой будет поставляться специальный файл справочной системы (MAN-файл). Понятно, если разработчик недобросовестный, он может и не создать такой файл, но это происходит очень редко.

Чтобы получить справку по какой-нибудь программе, нужно ввести команду:

```
man имя_программы
```

Вы никак не можете запомнить, как пишется та или иная команда? Если вы помните хотя бы, на какую букву она начинается, воспользуйтесь функцией автодополнения командной строки — введите первые буквы команды и нажмите комбинацию клавиш <Ctrl>+<D>. При первом нажатии система попытается дополнить команду. Иногда это невозможно — например, вы ввели `cl`. Ясное дело, в системе есть несколько команд, которые начинаются буквами "cl", и в такой ситуации система не может дополнить командную строку именем единственной команды, поэтому она выведет список доступных команд (рис. 8.4).



Рис. 8.4. Автодополнение в действии

Подробно работе с консолью посвящена вся *третья часть* книги, а в этой главе мы рассмотрим лишь то, что необходимо вам прямо сейчас для работы с системой.

8.3. Виртуальные консоли

При входе в систему выводится номер виртуальной консоли:

FreeBSD/i386 (denhost.localdomain) (ttyv0)

В нашем случае имя виртуальной консоли — ttyv0. Это первая консоль. Нумерация консолей начинается с 0. Для переключения между консолями служит комбинация клавиш <Alt>+<F*n*, где *n* — это номер консоли.

> ***Внимание!***
>
> Еще раз напомню, что нумерация консолей начинается с 0. Поэтому, когда вы нажимаете комбинацию <Alt>+<F1>, то переключаетесь на консоль ttyv0, нажатие <Alt>+<F2> приводит к переключению на консоль ttyv1, <Alt>+<F3> — на ttyv2 и т. д.

Когда запущен графический интерфейс (сервер X), консоли ttyv5, ttyv6 и ttyv7 резервируются за ним. Другими словами, нажав комбинацию клавиш <Alt>+<F6> (чтобы переключиться на консоль ttyv5), вы перейдете в графический режим. Чтобы переключиться обратно в консоль, используется комбинация клавиш <Ctrl>+<Alt>+<F*n*>, где *n* — номер консоли (не забывайте о нумерации с 0).

Виртуальные консоли — очень удобный механизм UNIX, позволяющий одновременно запускать в разных консолях несколько интерактивных программ. Например, на первой консоли у вас может быть запущен файловый менеджер mc, на второй — текстовый редактор (на этапе настройки системы редактировать файлы конфигурации придется часто), а на третьей — текстовый интернет-браузер.

8.4. Правильное завершение работы в системе

Для выхода из консоли (чтобы ею никто не воспользовался во время вашего отсутствия) предусмотрена команда `logout`.

Завершить работу системы можно командами:

- `reboot` — выполняет перезагрузку системы сразу после ввода команды;
- `halt` — завершает работу системы сразу после ввода команды;
- `shutdown` — завершает работу системы (в том числе и выполняет перезагрузку) в указанное время.

Самая "продвинутая" команда — `shutdown`, она, как уже было отмечено, позволяет завершить работу и перезагрузить систему в назначенное время. Предположим, что вы хотите уйти пораньше, но компьютер нужно выключить ровно в 19.30 (вдруг некоторые пользователи задержались на работе, а вы выключите сервер, — получится некрасиво). Вот тут-то вам и поможет команда `shutdown`:

```
# shutdown -h 20:30 [сообщение]
```

Если нужно завершить работу системы прямо сейчас, вместо времени укажите `now`:

```
# shutdown -h now
```

Для перезагрузки системы служит опция `-r`:

```
# shutdown -r now
```

Выключить питание вы сможете, когда увидите сообщение (рис. 8.5):

The operating system has halted.

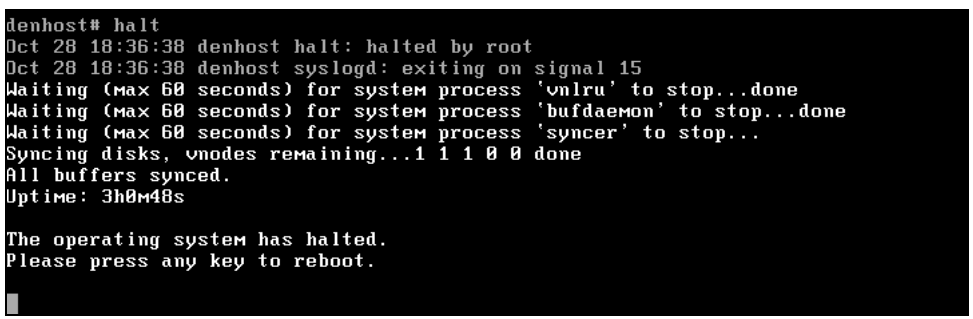


Рис. 8.5. Работа операционной системы завершена

8.5. Конфигуратор *sysinstall*

Для настройки системы некоторые администраторы предпочитают использовать конфигуратор `sysinstall` (рис. 8.6). Но знайте — все, что настраивается с помощью этого конфигуратора, можно сделать вручную путем редактирования конфигурационных файлов и ввода команд, что и будет продемонстрировано в этой книге далее.

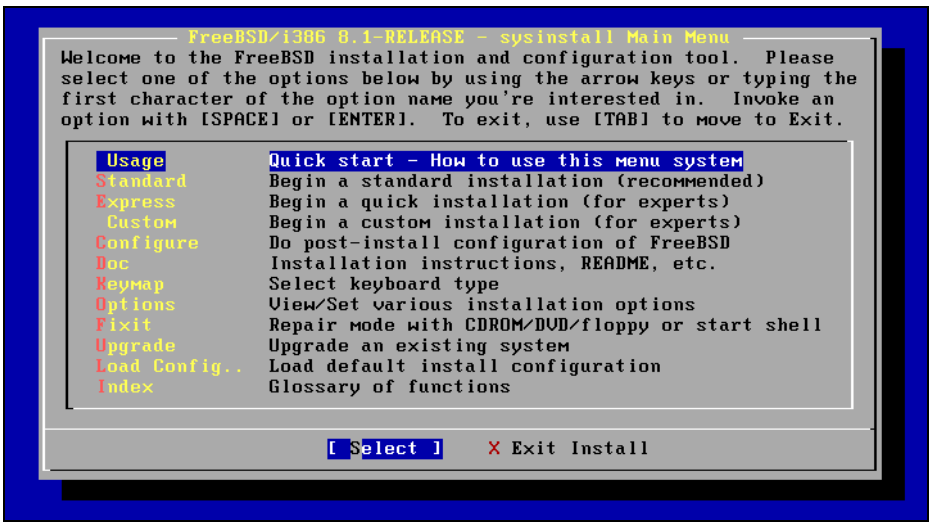


Рис. 8.6. Конфигуратор **sysinstall**

8.6. Файловый менеджер Midnight Commander

Команды для работы с файловой системой будут рассмотрены в *главе 14*, а пока, чтобы вам было удобнее "бороздить просторы" своего диска, установите файловый менеджер Midnight Commander (рис. 8.7).

> ***Примечание***
>
> Все мы помним классический двухпанельный файловый менеджер для DOS Norton Commander (nc). Позже на его базе были созданы другие подобные менеджеры: Volkov Commander, DOS Navigator, Midnight Commander и пр.

Если доступ к Интернету у вас настраивается по DHCP, и при установке системы вы выбрали настройку сети по DHCP, тогда просто введите команду `mc` (в *главе 14* вы найдете дополнительную информацию об этой команде):

```
# pkg_add -r mc
```

А вот если доступа к Интернету пока нет (потому что мы его еще не настроили), тогда придется потерпеть, поскольку пакета mc на дистрибутивном диске я не нашел.

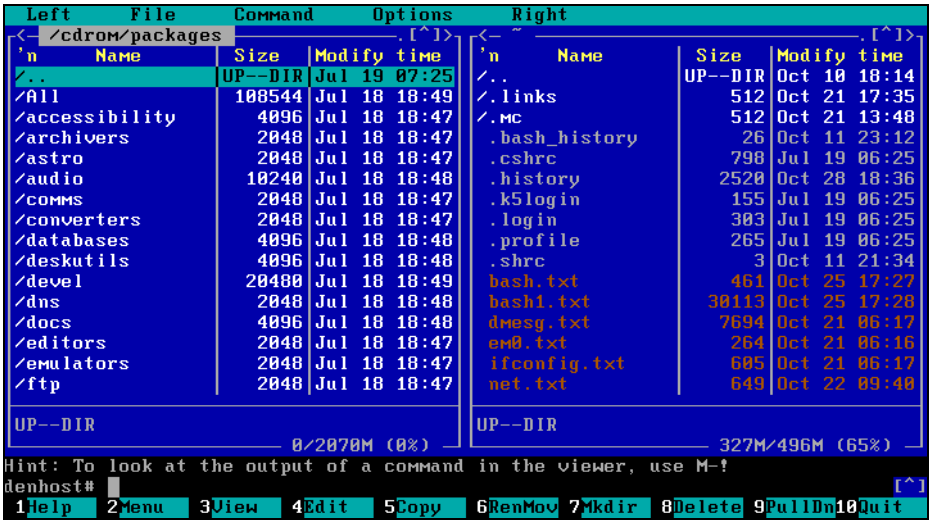


Рис. 8.7. Файловый менеджер **Midnight Commander**

8.7. Изменение редактора по умолчанию

В FreeBSD огромное значение имеет правильно выбранный редактор по умолчанию. Дело в том, что многие программы, редактирующие конфигурационные файлы (например, visudo, vipw) вызывают редактор по умолчанию. А по умолчанию используется старый (скорее, даже "древний") и не очень добрый (точнее, совсем неудобный) редактор vi. Могу поспорить, что если вы никогда не имели дела с UNIX, то даже не сможете выйти из этого текстового редактора, не зная, что нажимать. Нам, хотя мы и работаем в консоли, хочется иметь обычный удобный и более современный текстовый редактор. В FreeBSD такой есть — это редактор ee. Осталось только установить его по умолчанию, для чего нужно отредактировать файл ~/.cshrc:

```
# ee ~/.cshrc
```

Найдите в нем строку:

```
setenv EDITOR vi
```

Замените эту строку другой строкой:

```
setenv EDITOR ee
```

Нажмите клавишу <Esc>, чтобы увидеть меню редактора ee, нажмите клавишу <a> для выхода из редактора и еще раз клавишу <a> для сохранения файла. Запоминать, что нажимать, не нужно — вы увидите подсказки на каждом шагу (рис. 8.8). Чтобы изменения вступили в силу, введите команду `exit` и снова войдите в систему. Теперь вместо vi в вашей системе будет использоваться редактор ee.

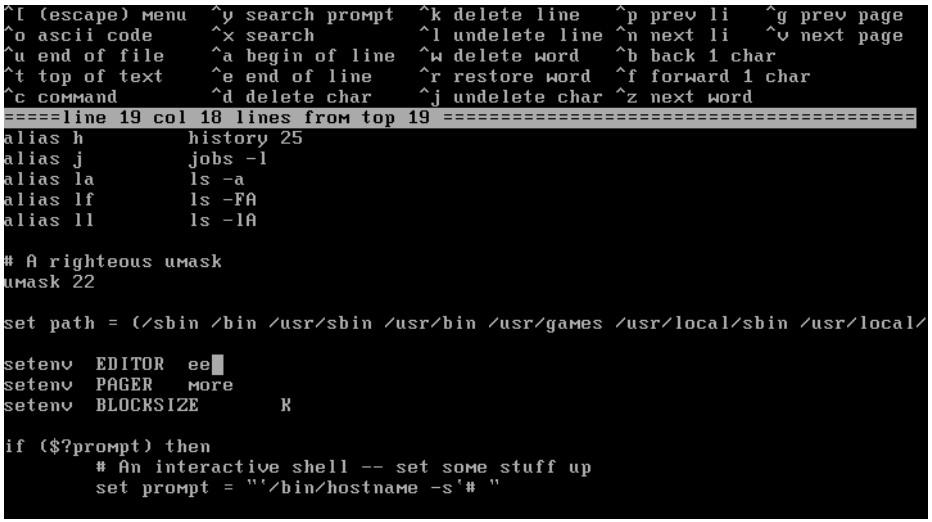


Рис. 8.8. Процесс редактирования файла ~/.cshrc

8.8. Использование редактора vi

Чуть ранее мы выбрали другой редактор — ee. Так зачем здесь информация о vi? Тому есть две причины. Во-первых, дабы читатели не обвинили меня в непрофессионализме. Мол, как это так — в книге по UNIX нет описания работы с "классикой"! Во-вторых, эта информация пригодится пользователям, уже запустившим ту же команду `vipw` для редактирования базы данных пользователей. Надо же как-то сохранить изменения? Не "убивать" же процесс только потому, что не знаешь, как из него выйти?

Со времен первых версий UNIX в современные системы перекочевал текстовый редактор vi (рис. 8.9). То, что ему больше тридцати лет — видно сразу. Более неудобного редактора я не видел! Согласен, что тогда это был прорыв, но сегодня редактор смотрится уж очень архаично.

Некоторые гурманы (я бы их назвал мазохистами) говорят, что к нему надо привыкнуть. Может и так, но сначала придется изучить длинное руководство (man) и выучить наизусть его команды. Как такового интерфейса пользователя в vi практически нет (можно сказать, что вообще нет) — а то, что есть, сложно назвать интерфейсом. Однако мы рассмотрим vi, хотя бы вкратце.

Редактор vi может работать в трех режимах:

- основной (визуальный) режим — в нем и осуществляется редактирование текста;
- командный режим — в нем осуществляется ввод специальных команд для работы с текстом (если сравнивать vi с нормальным редактором, то этот режим ассоциируется с меню редактора, где есть команды типа **Сохранить, Выйти** и т. д.);



Рис. 8.9. Текстовый редактор vi

- режим просмотра — используется только для просмотра файла (если надумаете использовать этот режим, вспомните про команду `less`).

После запуска редактора вы можете переключать режимы (как, будет сказано позже), но выбрать режим можно и при запуске редактора:

```
vi файл
vi -e файл
vi -R файл
```

Первая команда запускает vi и загружает файл. Вторая команда запускает vi в командном режиме и загружает файл. Третья команда — это режим просмотра файла. Если указанный файл не существует, то он будет создан. По умолчанию активируется именно командный режим, поэтому в ключе `-e` нет смысла.

После запуска vi главное знать, как из него выйти, — ведь в нем не будет привычной строчки меню, редактор также не будет реагировать на привычные комбинации клавиш вроде <Alt>+<X> или <Ctrl+<C>.

Основные команды редактора vi приведены в табл. 8.1.

Таблица 8.1. *Основные команды редактора vi*

Команда	Описание
`:q!`	Выход без сохранения
`:w`	Сохранить изменения
`:w <файл>`	Сохранить изменения под именем <файл>
`:wq`	Сохранить и выйти
`:q`	Выйти, если нет изменений

Таблица 8.1 *(окончание)*

Команда	Описание
i	Перейти в режим вставки символов в позицию курсора
a	Перейти в режим вставки символов в позицию после курсора
o	Вставить строку после текущей
O	Вставить строку над текущей
x	Удалить символ в позицию курсора
dd	Удалить текущую строку
u	Отменить последнее действие

Команды, которые начинаются с двоеточия, будут отображены в нижней строке, остальные просто выполняются, но не отображаются. Как уже было отмечено, у редактора vi есть два основных режима (режим просмотра не считается) — режим команд и режим редактирования (визуальный). Переключение в режим команд осуществляется нажатием клавиши <Esc>. Нажатие клавиш <i>, <a> и других переключает редактор в режим вставки, когда набираемые символы трактуются именно как символы, а не как команды. Для переключения обратно в командный режим служит клавиша <Esc>. В некоторых случаях (например, когда вы пытаетесь передвинуть курсор левее первого символа в строке) переход в командный режим осуществляется автоматически.

Теперь немного практики. Введите команду:

```
$ vi file.txt
```

Далее нажмите клавишу <i>, чтобы переключиться в режим вставки. Наберите любой текст, но постарайтесь не ошибаться, поскольку исправление ошибок в vi — дело, требующее отдельного разговора.

Затем нажмите клавишу <Esc> и введите `:wq`. После выхода из редактора введите команду:

```
cat file.txt
```

Так вы убедитесь, что файл создан и в него сохранен введенный вами текст.

Продолжим рассмотрение редактора. Если нажать не клавишу <i>, а клавишу <a>, то вы тоже перейдете в режим вставки, но с одним отличием — введенный текст будет вставляться не перед символом, в котором находится курсор, а после него. Также в режим вставки можно перейти нажатием клавиш <o> или <O>. В первом случае будет добавлена пустая строка после текущей строки, а во втором — перед текущей строкой, и весь дальнейший ввод будет восприниматься именно как ввод текста, а не команд.

Чтобы удалить символ, нужно перейти в режим команд и над удаляемым символом нажать клавишу <x>. Да, клавиши <Backspace> и <Delete> тут не работают. Точнее, <Backspace> работает, но для удаления последней непрерывно введенной последовательности символов. Например, у нас есть текст "vi — текстовый редактор". Вы перейдете в режим вставки и измените текст так "vi — неудобный тексто-

вый редактор". Нажатие клавиши <Backspace> удалит слово "неудобный", но не сможет удалить тире и другие символы.

Чтобы удалить строку, в которой находится курсор, нужно нажать клавиши <d>+<d>. Помните, что vi считает строкой не то, что вы видите на экране, а последовательность символов до первого символа новой строки (\n). Если строка длиннее 80 символов, то она переносится на две экранных строки и визуально выглядит как две строки, а не как одна.

Чтобы перейти в конец строки (клавиши <Home> и <End> тоже не работают, как вы успели заметить, если уже запускали vi), нужно нажать клавишу <$>. При навигации курсор перемещается не по экранным линиям, а как раз по строкам текста.

Для отмены последней операции служит клавиша <u>. Вот только истории изменений нет, да и по клавише <u> отменяется вся предыдущая команда целиком. Например, вы создали файл, перешли в режим вставки (клавиша <i>) и ввели весь текст большой медицинской энциклопедии. Если вы нажмете клавишу <u>, то она отменит всю предыдущую команду, то есть удалит весь введенный вами текст. Так что, будьте осторожны.

Азы vi я вам преподнес. Но не думаю, что вы будете ним пользоваться. Если есть желание продолжить знакомство, введите команду:

```
man vi
```

8.9. Русификация консоли

Перед тем как перейти к следующей главе, русифицируем нашу консоль. Первым делом нужно отредактировать файл /etc/rc.conf. Для этого я воспользуюсь встроенным редактором mc (мне так удобнее), а вы можете использовать редактор ее. В файл /etc/rc.conf добавьте следующие строки (рис. 8.10):

```
keymap="ru.koi8-r.win"
font8x14="koi8-r-8x14"
font8x16="koi8-r-8x16"
font8x8="koi8-r-8x8"
keyrate="fast"
```

Строка выбора раскладки (`keymap`) в этом файле, скорее всего, уже есть, но ее нужно изменить, как здесь показано. Мы будем использовать кодировку символов KOI8-R, но раскладку клавиш Windows, — так нам привычнее. Далее идут строки выбора шрифтов, а последняя строка заставляет нашу клавиатуру работать быстрее.

После редактирования файла rc.conf нужно отредактировать файл /etc/passwd, но делать это следует с помощью команды `vipw`. Перед вызовом этой команды установите по умолчанию текстовый редактор ее *(см. разд. 8.7)* — так и вам станет проще, и иллюстрации в книге будут соответствовать реальности.

В пятое поле базы данных пользователей (где описывается класс регистрации) нужно вставить значение `russian`, например:

```
denis:пароль:1001:1001:russian:0:0:Denis:/home/denis:/bin/sh
```

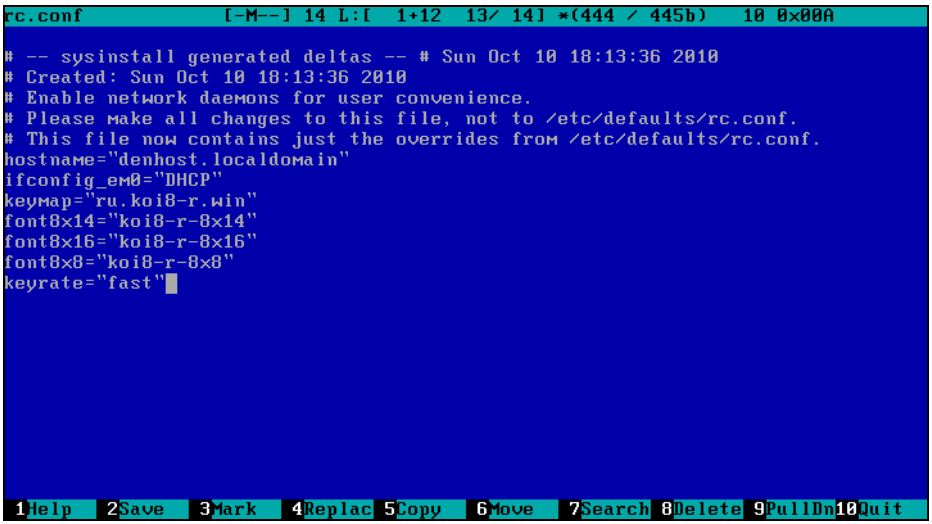


Рис. 8.10. Редактирование файла /etc/rc.conf

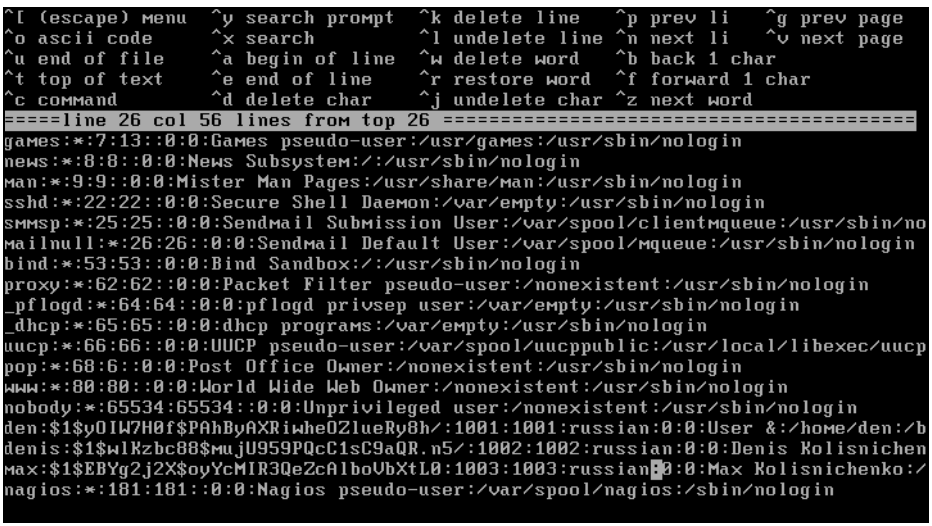


Рис. 8.11. Утилита vipw

Процесс редактирования этого файла изображен на рис. 8.11. Изменения нужно произвести для всех пользователей системы (не забудьте о пользователе root!).

Следующий шаг — это редактирование файла /etc/ttys. Найдите в этом файле строки, задающие параметры консоли:

```
ttyv0   "/usr/libexec/getty Pc"         cons25        on  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25        on  secure
```

```
ttyv2   "/usr/libexec/getty Pc"         cons25         on  secure
ttyv3   "/usr/libexec/getty Pc"         cons25         on  secure
ttyv4   "/usr/libexec/getty Pc"         cons25         on  secure
ttyv5   "/usr/libexec/getty Pc"         cons25         on  secure
ttyv6   "/usr/libexec/getty Pc"         cons25         on  secure
ttyv7   "/usr/libexec/getty Pc"         cons25         on  secure
```

Измените для консолей ttyv0 — ttyv7 тип консоли с `cons25` на `cons25r` (см. рис. 8.12):

```
ttyv0   "/usr/libexec/getty Pc"         cons25ron  secure
# Virtual terminals
ttyv1   "/usr/libexec/getty Pc"         cons25ron  secure
ttyv2   "/usr/libexec/getty Pc"         cons25ron  secure
...
```

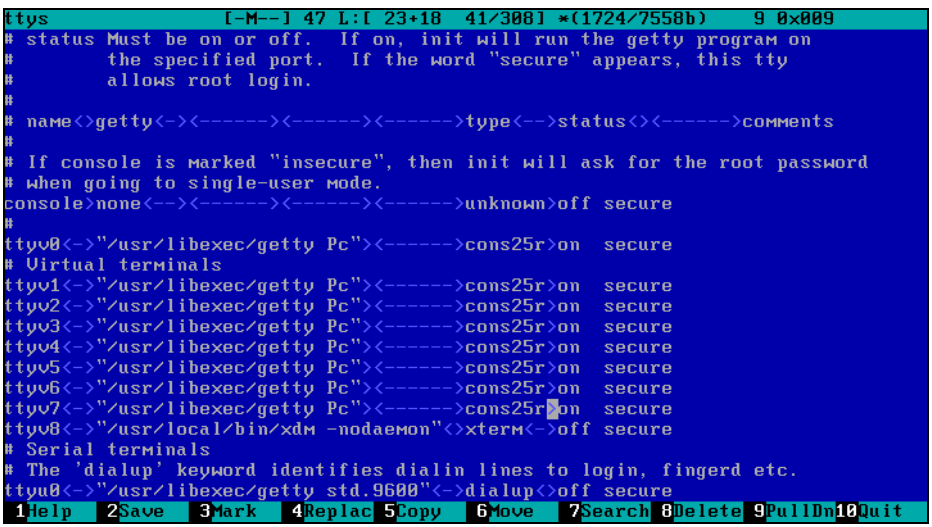


Рис. 8.12. Редактирование файла /etc/ttys

Примечание

Сразу после типа консоли следует статус консоли. Параметр `on` означает, что консоль включена, но вы можете задать параметр `off` для выключения консоли. Параметр `secure` означает, что с этой консоли возможен вход пользователя root.

Почти все. Осталось в файле ~/.profile изменить значение переменной `TERM`, хотя это делать необязательно (рис. 8.13):

```
TERM=${TERM:-cons25r}
```

Теперь надо перезапустить сервис syscons (хотя я перезагрузил всю машину командой `reboot`):

```
# /etc/rc.d/syscons restart
```

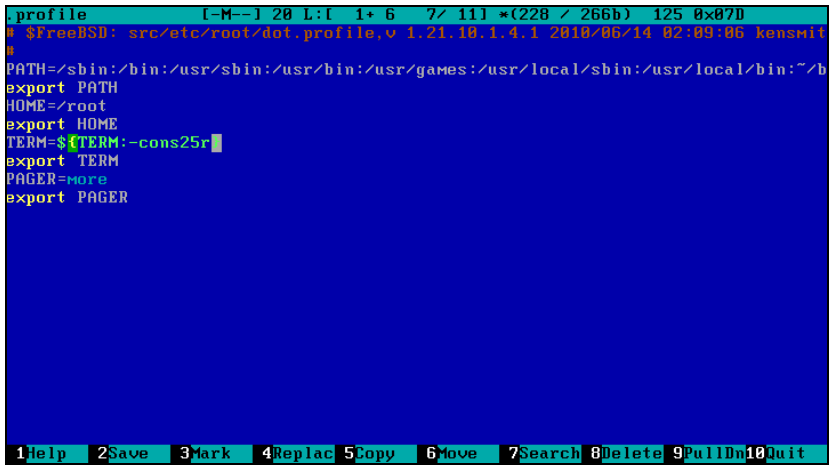


Рис. 8.13. Изменение файла ~/.profile

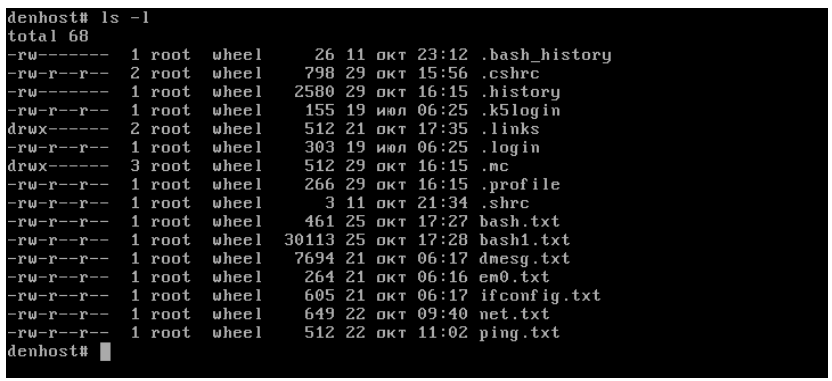


Рис. 8.14. Команда ls русифицирована

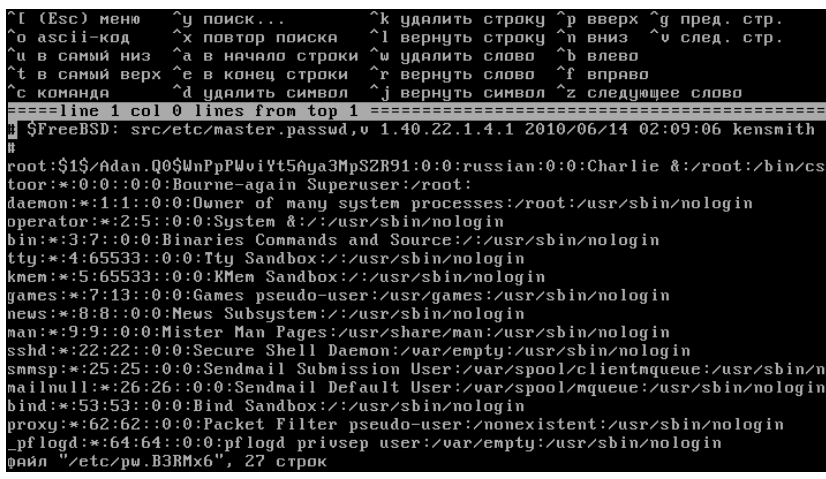


Рис. 8.15. Текстовый редактор ее русифицирован

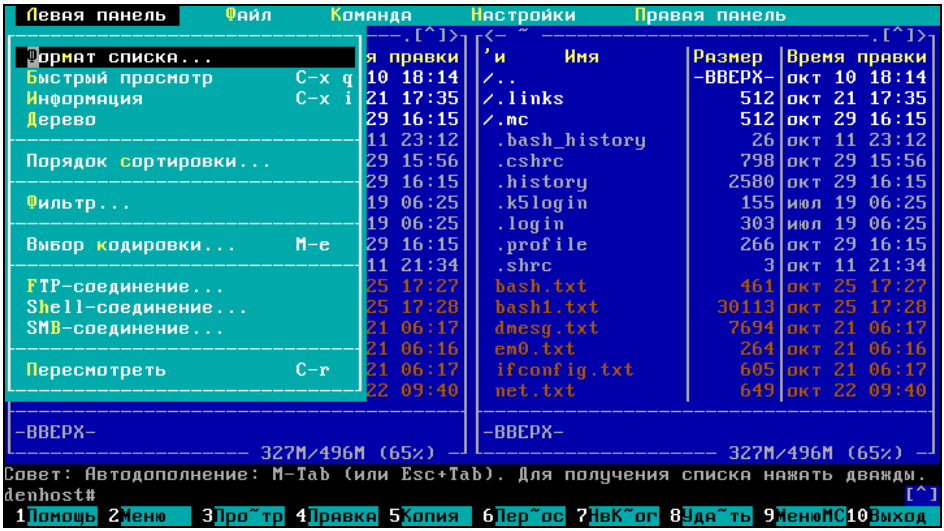


Рис. 8.16. Файловый менеджер mc русифицирован



Рис. 8.17. Ввод русского текста

После перезагрузки мы получаем полностью русифицированную консоль. Введите команду `ls -l` — вывод команды `ls` будет на русском языке (рис. 8.14), меню редактора ее тоже будет на русском (рис. 8.15), даже мой любимый файловый менеджер Midnight Commander русифицировался (рис. 8.16). Так что русификация консоли — дело полезное. Чуть не забыл — переключение на русский язык ввода осуществляется с помощью комбинации <Ctrl>+<Shift> (рис. 8.17).

8.10. Переход на UTF-8

В предыдущем разделе мы русифицировали систему, используя кодировку KOI8-R — это классическая кодировка русского языка всех UNIX-систем. Но за кодировкой UTF-8 (Юникод) — будущее, и чем раньше мы на нее перейдем, тем лучше. Соответственно, после перехода на UTF-8 проблем с русским, как и с другими языками, у вас не будет — в этом и заключается преимущество UTF-8.

Переходить на UTF-8 или нет, каждый решает сам. Одно могу сказать: FreeBSD полностью поддерживает UTF-8, поэтому практически никаких подводных камней у вас не будет (почему "практически" — см. в конце раздела).

Для перехода на UTF-8 первым делом нужно изменить класс `russian`. Откройте файл /etc/login.conf и найдите в нем строки:

```
russian|Russian Users Accounts:\
          :charset=KOI8-R:\
          :lang=ru_RU.KOI8-R:\
          :tc=default:
```

Это и есть описание класса `russian` — для учетных записей русскоязычных пользователей. Отредактируйте класс `russian` так:

```
russian|Russian Users Accounts:\
          :charset=UTF-8:\
          :lang=ru_RU.UTF-8:\
          :tc=default:
```

Чтобы изменения вступили в силу, после сохранения файла введите команду:

```
# cap_mkdb /etc/login.conf
```

Удалите из файла rc.conf все, что связано с KOI8-R. После этого отредактируйте файл ~/.xinitr (если вы используете графический интерфейс) — добавьте в него строки:

```
export LANG='ru_RU.UTF-8'
export LC_ALL='ru_RU.UTF-8'
```

Собственно, вот и весь переход. Но после перезагрузки вы, все же, столкнетесь с двумя небольшими подводными камнями. Во-первых, обнаружится, что редактор ее не поддерживает Юникод, поэтому придется установить редактор nano:

```
# pkg_add -r nano
```

Второй подводный камень вы обнаружите, если к этому времени успели создать файлы, содержащие в именах русские символы. Поскольку использовалась кодировка KOI8-R, вместо русских символов после перехода на UTF-8 вы увидите вопросительные знаки. Нужно перекодировать все имена файлов из кодировки KOI8-R в UTF-8. Для этого мы применим утилиту `convmv`:

```
# convmv -r -f KOI8-R -t UTF-8 /root/*
```

Теперь все имена файлов и каталогов в каталоге /root будут перекодированы.



Глава 9

Установка графической среды GNOME

9.1. Графический интерфейс в FreeBSD

Поскольку обычно FreeBSD применяется на сервере, то графический интерфейс пользователя ей не нужен — для администратора вполне хватает и консоли. Но некоторые пользователи, немного поработав с этой операционной системой, становятся ее фанатами и устанавливают FreeBSD на любые компьютеры вплоть до личного ноутбука. А на таких компьютерах уже должен быть графический интерфейс. Да, существует текстовый браузер links, но Web-страницы гораздо удобнее просматривать браузером Mozilla в графическом режиме.

Не буду затягивать изложение и рассказывать об эволюции графических интерфейсов для UNIX. Скорее всего, вы уже знакомы с Linux и знаете, что к чему. Для тех же, кто впервые увидел UNIX, сделаю небольшое введение.

Графическая подсистема UNIX появилась еще в далеком 1984 году (да, значительно раньше, чем Windows с ее окошками) и называлась она тогда X Window System (или просто X Window). В 2004 году X Window переименовали в X.Org.

Сама по себе X Window (точнее, X.Org) не предоставляет графического интерфейса, то есть не определяет вид окошек, меню, рабочего стола, способ переключения между окнами рабочего стола и т. д. Для этого служит другая программа — *оконный менеджер*. Для UNIX было создано множество оконных менеджеров, но самыми удачными среди них являются GNOME (GNU Network Object Model Environment) и KDE (K Desktop Environment). К настоящему моменту оба эти продукта переросли само определение оконного менеджера. KDE и GNOME не только определяют внешний вид интерфейса пользователя, но и предоставляют API для создания работающих в них программ. Поэтому KDE и GNOME принято называть *графическими средами*.

Какую графическую среду выбрать — GNOME или KDE — решает, как правило, сам пользователь. Обе хороши, и выбор в большинстве случаев зависит от личных предпочтений. Мне больше нравится GNOME, поэтому эта графическая среда и будет здесь рассмотрена. Точнее, будет рассмотрена только установка GNOME — работать в GNOME сможет даже ребенок, а вот установка заслуживает отдельного разговора.

О графической среде KDE скажу только, что она потребляет больше системных ресурсов, поэтому на старых компьютерах, где установлено 512 Мбайт оперативной

памяти или меньше, лучше использовать GNOME. Это раньше, во времена KDE версии 3.5, эта графическая среда нормально работала на компьютерах с 512 (и даже меньше) Мбайт оперативной памяти. Но для работы с современной версией KDE 4 предпочтительнее иметь ОЗУ не менее 1 Гбайт. Для современных компьютеров это не проблема, а вот для компьютеров, собранных 3–4 года назад, порой сложно даже найти модуль оперативной памяти — при всем желании его купить.

Примечание

Уже отмечалось ранее, что при написании этой книги я следовал одному простому правилу: не дублировать "хэндбук" — Руководство, написанное командой разработчиков FreeBSD. Данная глава ни строчкой не дублирует "хэндбук", а в некоторых местах даже дополняет его.

9.2. Установка портов и пакетов

В Интернете есть много рекомендаций по установке GNOME в FreeBSD, но, знакомясь с ними, обращайте внимание на дату публикации. Вряд ли статьи 5–6-летней давности будут сегодня актуальны.

Как сказано на сайте **http://www.freebsd.org/gnome/**, самый простой способ обзавестись GNOME — это установить ее из *порта* (исходного кода программы) или из *пакета* x11/gnome2 (способы установки программного обеспечения в FreeBSD описаны в *главе 18*). Для экономии времени будем устанавливать GNOME из пакетов:

```
# pkg_add -r xorg
# pkg_add -r gnome2
```

Примечание

При использовании оболочки tcsh *(см. разд. 11.3)* не забудьте после второй команды ввести еще команду `rehash`.

А теперь надо запастись терпением. Первая команда устанавливает графическую подсистему X.Org, а вторая — графическую среду GNOME. Но прежде системе нужно получить указанные пакеты из Интернета, распаковать их и уже потом установить. Даже не представляю, сколько времени понадобится, чтобы собрать все из портов… Если под рукой есть дистрибутивный DVD, можно сэкономить немного времени, установив все пакеты с него. Для этого DVD нужно смонтировать:

```
# mount /cdrom
# cd /cdrom/packages/All
# pkg_add xorg
# pkg_add gnome2
```

Внимание!

В случае установки пакетов с DVD не следует использовать опцию `-r` команды `pkg_add`.

Сам же я устанавливаю пакеты из Интернета по двум причинам: есть надежда заполучить самые последние версии пакетов и нужно же как-то использовать канал

в Интернет со скоростью 80 Мбит/с? Так что мне по большому счету все равно, откуда загружать пакеты: с DVD или из Интернета.

Советую далеко от компьютера не отходить — при установке пакетов выводятся различные информационные сообщения. Вы узнаете много интересного, например, какой модуль ядра нужен для webcamd (рис. 9.1) и прочее в этом же духе. Через некоторое время вы увидите заветное сообщение — поздравление с установкой GNOME (рис. 9.2).

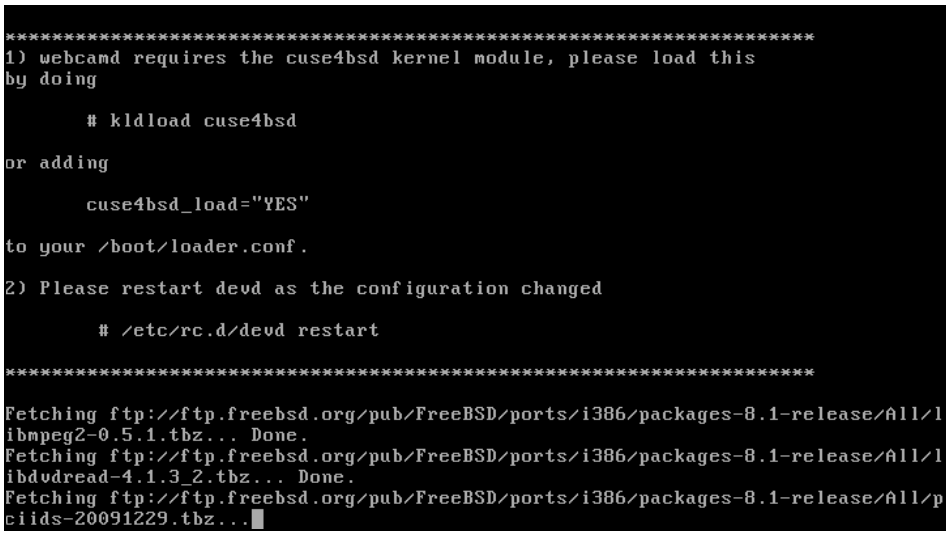


Рис. 9.1. Процесс установки GNOME

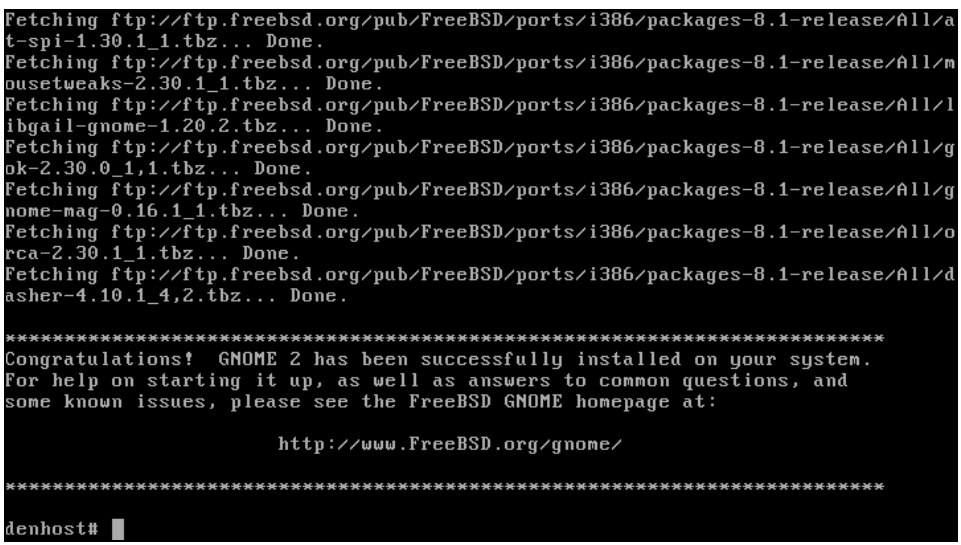


Рис. 9.2. Установка GNOME завершена

Что делать дальше? Тут возможны варианты. Вы можете сразу приступить к настройке X.Org или установить дополнительные пакеты, а уже потом настраивать X.Org. При этом надо иметь в виду, что пока вы не настроите X.Org, запуск GNOME будет невозможен.

Я предлагаю приступить сразу к настройке X.Org, а потом уже доустановить (если они вам нужны) следующие пакеты:

- gnome2-fifth-toe — базовые приложения GNOME;
- gnome2-power-tools — утилиты для администратора;
- gnome2-office — офисные приложения GNOME;
- gnome2-hacker-tools — приложения для разработчиков.

9.3. Настройка запуска GNOME

Первым делом добавим в файл /etc/rc.conf строки, обеспечивающие запуск сервисов, необходимых для работы X.Org:

```
hald_enable="YES"
dbus_enable="YES"
```

Чтобы после этого не перезагружать машину, запустим эти сервисы в первый раз вручную:

```
# /usr/local/etc/rc.d/hald start
# /usr/local/etc/rc.d/dbus start
```

В большинстве случаев X.Org может запуститься без всяких настроек (то есть без редактирования конфигурационного файла). Проверим это — просто введите команду: `# startx`

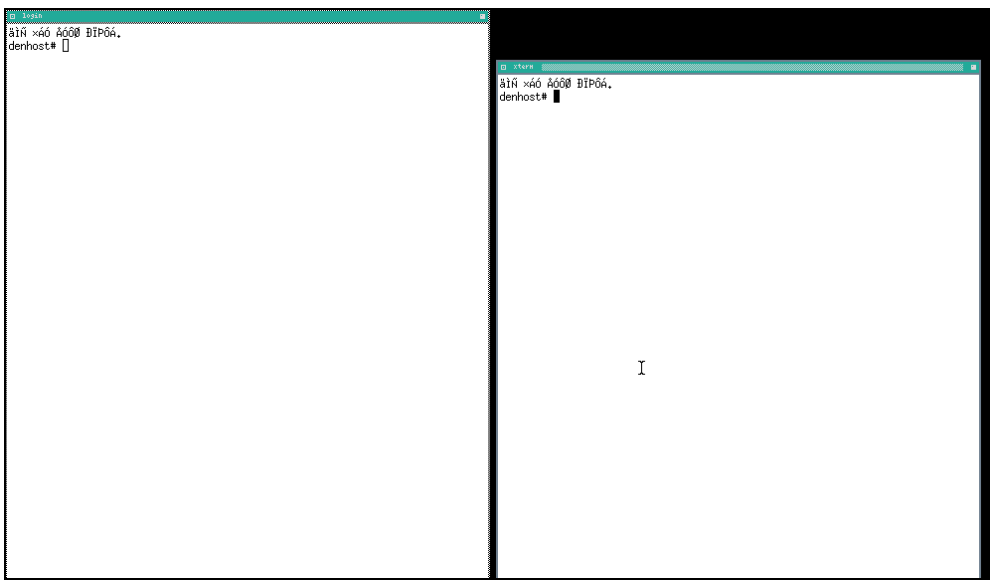


Рис. 9.3. Первый запуск X.Org

Лично у меня все запустилось с первого раза (рис. 9.3).

Примечание

Изображение на рис. 9.3 полностью соответствует действительности — именно это вы увидите на экране после ввода команды `startx`: два терминала и ничего больше. Не впечатляет? Чтобы обеспечить запуск GNOME, еще придется потрудиться, но сама X.Org уже работает.

Для запуска GNOME введите в терминале xterm команду:

```
# gnome-session
```

Начнется запуск GNOME (рис. 9.4) и через некоторое время вы увидите ее рабочий стол (рис. 9.5). Для нас сейчас самое главное, что GNOME запускается и что ее не нужно локализовать — все уже сделано за нас. Теперь завершите сеанс GNOME (нужная команда находится в меню **Система**).

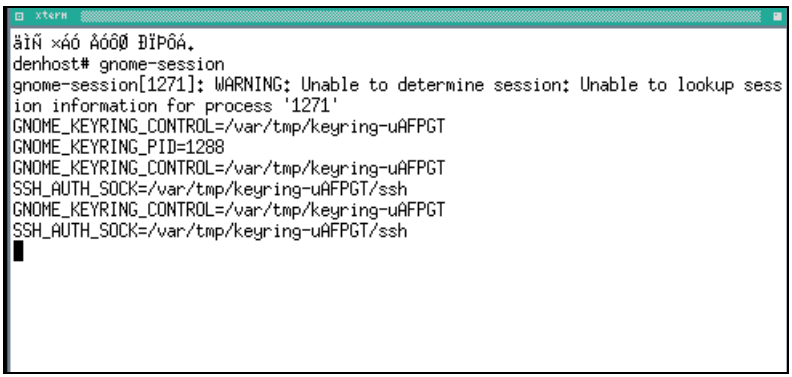


Рис. 9.4. Процесс запуска GNOME

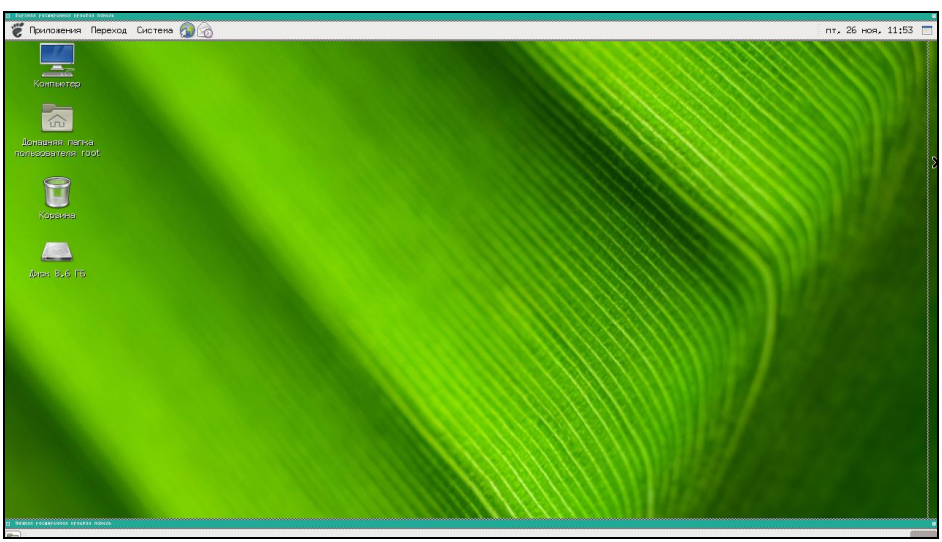


Рис. 9.5. GNOME запущена

Примечание

Изображение на рис. 9.5 тоже не очень красивое — как будто GNOME запущена в окне какого-то оконного менеджера. По сути, так оно и есть — именно это вы увидите при первом запуске GNOME. Я мог бы вместо этой иллюстрации поставить скриншот нормально настроенной среды GNOME, но поместил эту, чтобы иллюстрации в книге максимально соответствовали тому, что вы увидите на экране, если будете следовать моим инструкциям. Далее, по мере настройки, вы увидите разницу — GNOME будет выглядеть нормально, то есть так, как она выглядит в Linux.

Теперь нужно решить, как вы будете запускать GNOME. Можно запускать ее автоматически при запуске системы. В этом случае вы увидите графический менеджер входа в систему (GNOME Display Manager, GDM), введете свое имя пользователя и пароль, после чего откроется рабочий стол GNOME. Если вы когда-нибудь работали с Linux, то понимаете, о чем я говорю, — графический интерфейс в Linux загружается автоматически.

Для этого варианта, как написано в "хэндбук", нужно добавить в файл /etc/rc.conf строку:

```
gdm_enable="YES"
```

Откройте "хэндбук" по адресу **http://www.freebsd.org/doc/ru/books/handbook/x11-wm.html**. На указанной страничке написано, что достаточно добавить заветную строку в файл /etc/rc.conf и больше ничего делать не нужно. Хочется процитировать этот совет.

Цитата из "хэндбука"

Самый простой путь запустить GNOME — это использовать GDM (GNOME Display Manager). GDM, который устанавливается как часть GNOME (но отключен по умолчанию), может быть включен путем добавления gdm_enable="YES" в /etc/rc.conf. После перезагрузки GNOME запустится автоматически после того, как вы зарегистрируетесь в системе. Никакой дополнительной конфигурации не требуется.

Хорошо, следуем указаниям Руководства. Но там ни слова не сказано, что добавить ту самую строчку нужно после загрузки сервисов hald и dbus:

```
hald_enable="YES"
dbus_enable="YES"
gdm_enable="YES"
```

И если сделать иначе, у вас ничего не получится.

Итак, добавили строчку запуска GDM, перезагружаем компьютер и видим экран входа в систему — GDM (рис. 9.6). Впрочем, то, что мы видим, нельзя назвать экраном входа — это просто заставка с фоновым рисунком и именем компьютера. Более толку от нее нет, потому что в такой конфигурации GDM не позволяет войти в систему.

Почему-то в Руководстве забыли упомянуть, что нужно подключить псевдофайловую систему /proc. Нажмите комбинацию клавиш <Ctrl>+<Alt>+<F1> (так вы перейдете в консоль), войдите как root и введите команду:

```
# ee /etc/fstab
```

Добавьте в этот файл строку:

```
proc     /proc     procfs     rw     0  0
```

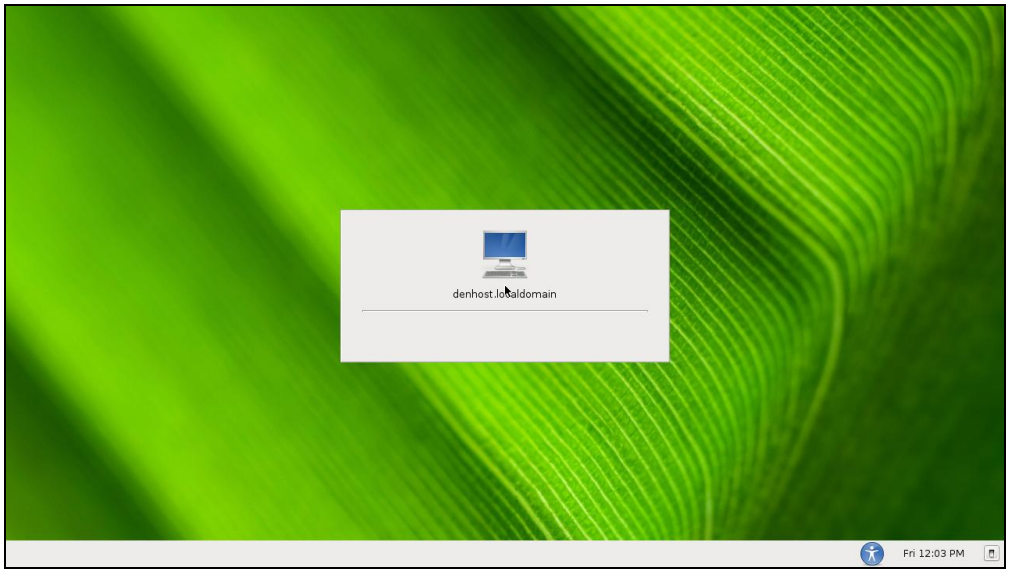


Рис. 9.6. Запущен GDM

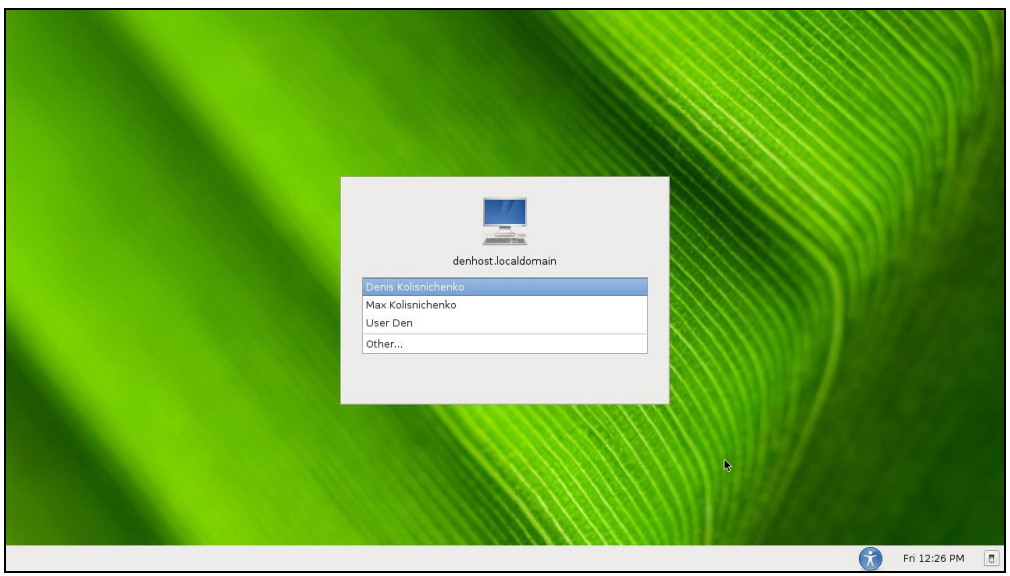


Рис. 9.7. Выбор пользователя в GDM

Сохраните файл и перезагрузите компьютер: `# reboot`

После перезагрузки все пойдет нормально: сначала нужно будет выбрать пользователя (рис. 9.7), затем ввести его пароль (рис. 9.8). Чтобы войти как root, выберите опцию **Other** (см. рис. 9.7), затем введите имя пользователя `root`, а затем и пароль root.

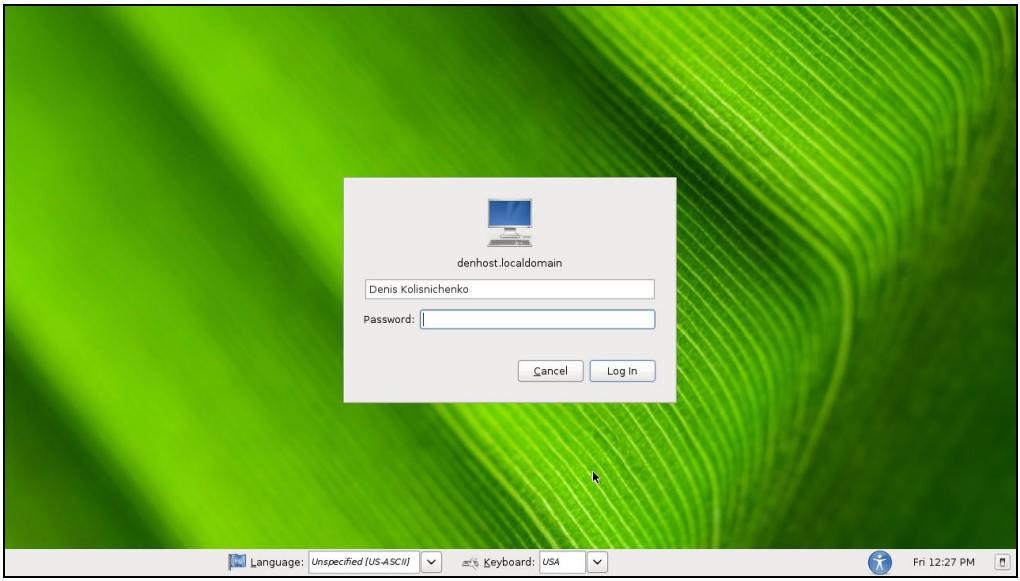


Рис. 9.8. Ввод пароля



Рис. 9.9. Интерфейс GNOME на английском языке

Если класс пользователя, вошедшего в систему с помощью GDM, отличен от `russian` *(см. разд. 8.9)*, интерфейс GNOME будет английским (рис. 9.9). Если же класс пользователя равен `russian`, интерфейс окажется локализованным (рис. 9.10). О русификации самого GDM можно прочитать в последнем разделе этой главы.

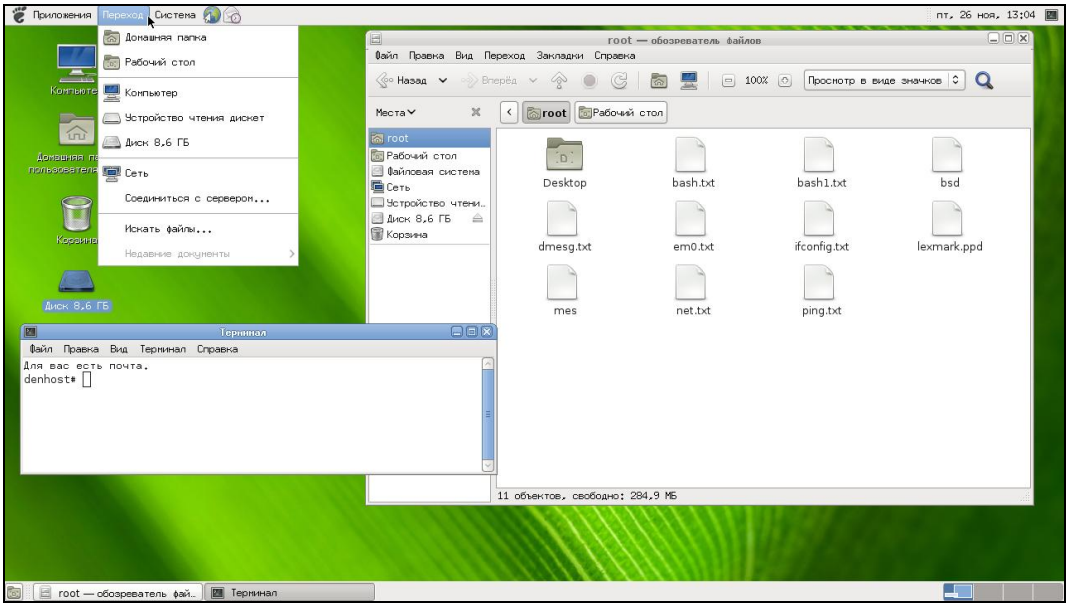


Рис. 9.10. Интерфейс GNOME на русском языке

Теперь рассмотрим второй способ запуска GNOME. Лично мне, если система установлена на сервере, не нравится, когда GDM загружается при запуске компьютера. Логика проста: FreeBSD установлена на сервере, с которым я обычно работаю удаленно, — по SSH. Следовательно, автоматически запущенный GDM будет только даром потреблять системные ресурсы. Когда же я начну работать за консолью сервера (имеется в виду непосредственно перед монитором и клавиатурой), тогда я введу команду `startx` для запуска GNOME. И мне удобнее, когда при вводе команды `startx` сразу запустится среда GNOME, — чтобы не вводить каждый раз еще и команду `gnome-session`.

Чтобы реализовать такой способ запуска, достаточно добавить команду `gnome-session` в файл ~/.xinitrc:

```
echo "/usr/local/bin/gnome-session" > ~/.xinitrc
```

Это действие нужно проделать для каждого пользователя, который будет работать в графическом режиме.

9.4. Несколько слов о русификации системы

9.4.1. Добавление русской раскладки

Перед тем как перейти к следующей главе, расскажу, как добавить русскую раскладку. Впрочем, вы должны это знать, если хотя бы раз в жизни работали с GNOME. Однако, если это не так, выполните команду меню **Система | Параметры | Клавиатура**. В открывшемся окне (рис. 9.11) нажмите кнопку **Добавить,**

выберите русскую раскладку (рис. 9.12) и нажмите кнопку **Добавить**. Русская раскладка будет добавлена в список (рис. 9.13).

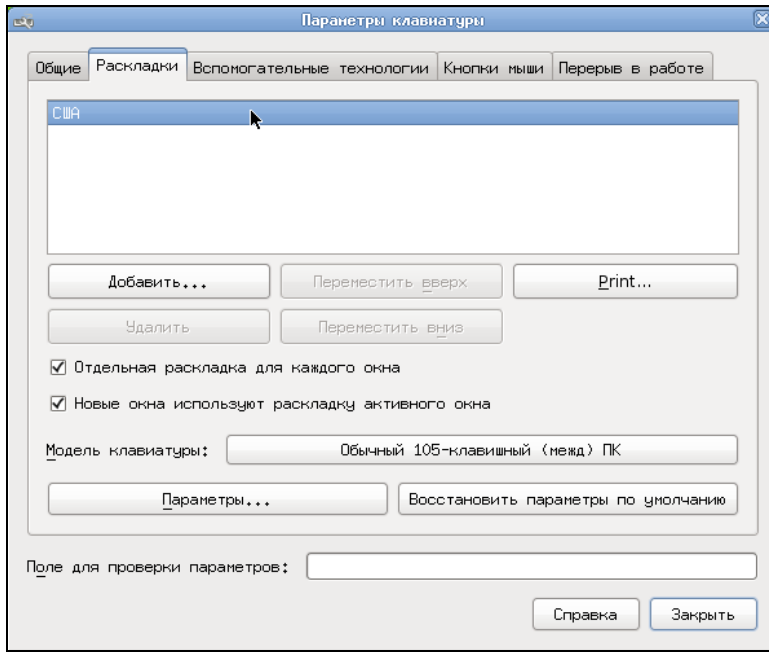


Рис. 9.11. Параметры клавиатуры

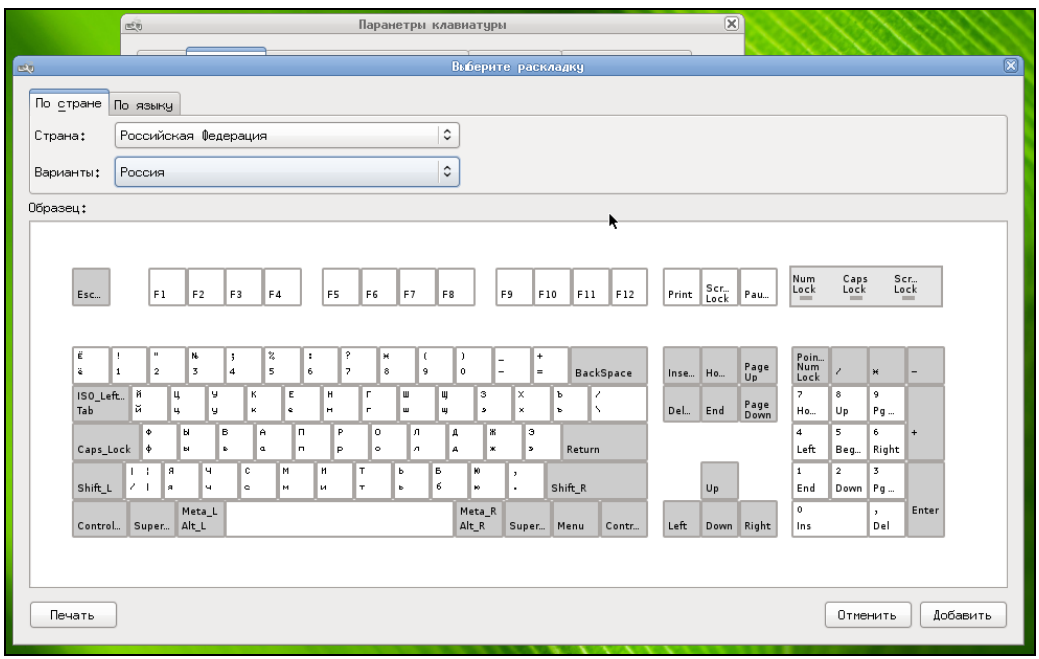


Рис. 9.12. Выбор русской раскладки

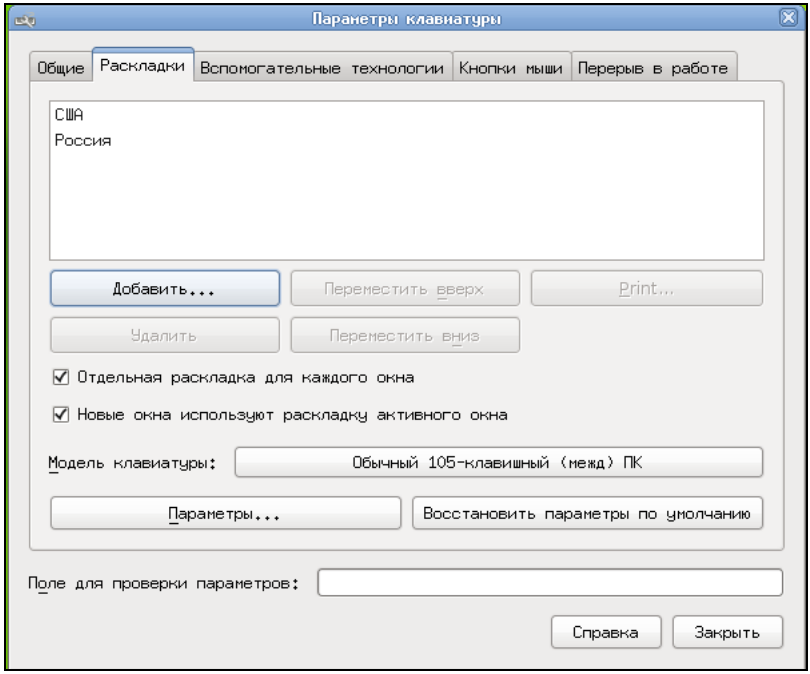


Рис. 9.13. Русская раскладка добавлена

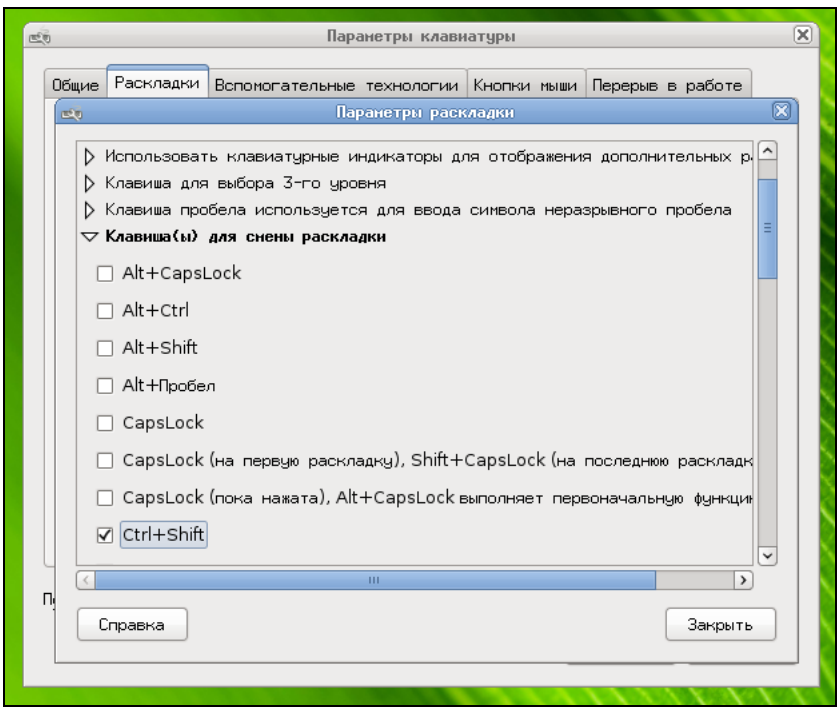


Рис. 9.14. Выбор комбинации клавиш для смены раскладки

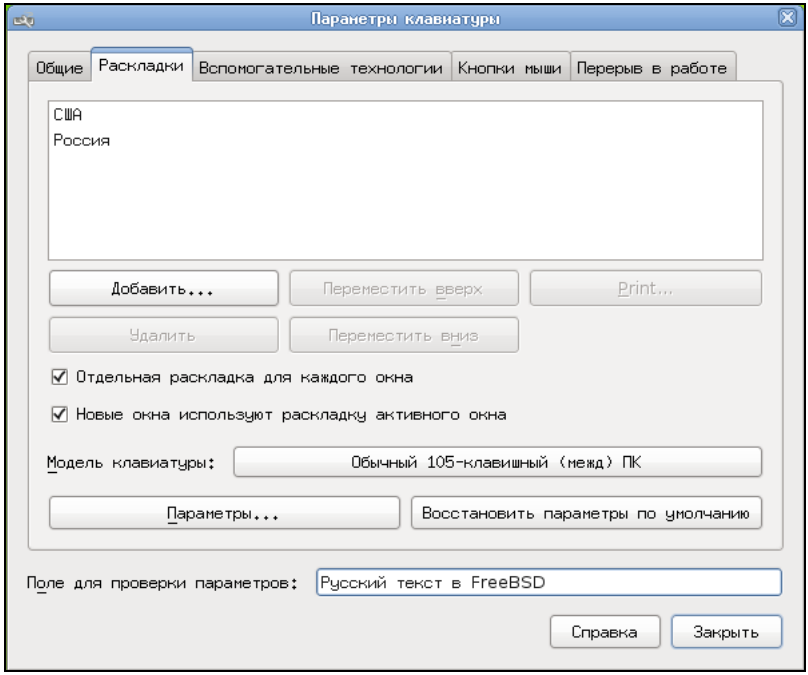


Рис. 9.15. Русская раскладка работает

Теперь определим комбинацию клавиш, которая будет использоваться для переключения раскладок. Нажмите кнопку **Параметры** и в группе **Клавиша (и) для смены раскладки** (рис. 9.14) выберите раскладку, которая вам больше нравится. Нажмите кнопку **Закрыть**, в поле **Поле для проверки параметров** (рис. 9.15) попробуйте ввести текст (только не забудьте переключиться на русский!).

9.4.2. О соответствии кодировок GNOME и консоли

Попробуйте в графическом терминале (**Приложения | Стандартные | Терминал**) создать файл, содержащий русские символы в имени, например:

```
# touch русский
```

Теперь выполните выход из GNOME. Если вы запускали GNOME командой `startx`, достаточно выбрать команду завершения сеанса из меню **Система**. Если GNOME запускалась посредством GDM, просто нажмите комбинацию клавиш <Ctrl>+<Alt>+<F1> и войдите в систему в консоли. Посмотрите, как в консоли отображается имя нашего файла. Если вы следовали инструкциям из *главы 8*, все будет нормально. А вот если вы воспользовались какими-либо другими советами или Руководством, у вас могут быть проблемы с кодировкой — вместо русского имени получится абракадабра. Все зависит от кодировки, которую вы выбрали при русификации системы. Поэтому или придется разбираться самостоятельно, или перенастроить систему в соответствии с инструкциями из *главы 8*.

Чтобы окно входа в систему (GDM) было на русском языке, в файл конфигурации /etc/rc.conf после строки загрузки GDM добавьте строку:

```
gdm_lang="ru_RU.KOI8-R"
```

После этого и GDM станет русским (рис. 9.16). Эта строка также установит локаль ru_RU.KOI8-R принудительно для всех пользователей, вошедших в GNOME.

> ***Внимание!***
>
> Название кодировки KOI8-R должно быть написано именно прописными буквами, иначе выбор кодировки не сработает.

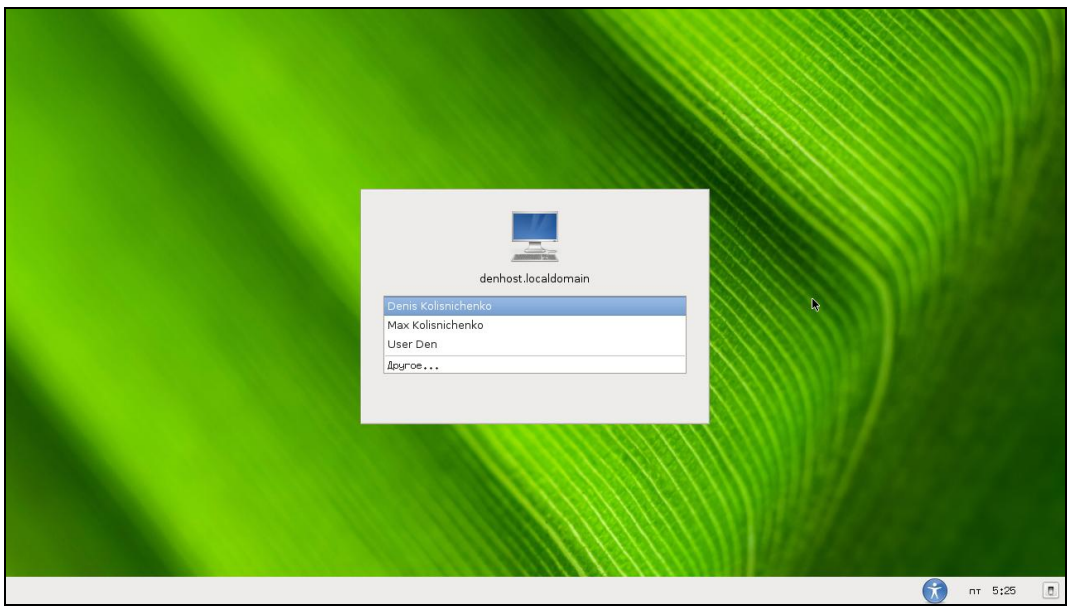


Рис. 9.16. GDM на русском языке

Кодировку KOI8-R мы выбрали, поскольку использовали ее в *главе 8*. Если у вас системная локаль использует другую кодировку, то вы должны указать ее в переменной `gdm_lang`. Например, для кодировки UTF-8 нужно использовать следующее значение:

```
gdm_lang="ru_RU.UTF-8"
```

Соответствие кодировок как раз и необходимо для нормальной работы с русскоязычными именами файлов. Если вы установите для GNOME кодировку UTF-8, а для консоли KOI8-R, GNOME тоже будет русифицирована, но возникнут проблемы при создании файлов с русскими буквами в имени. Так, в GNOME имя файла будет отображаться без проблем (потому что вы для GNOME использовали кодировку UTF-8), а в консоли выйдет абракадабра (потому что в ней использована кодировка KOI8-R).



Глава 10

Тонкая настройка графической подсистемы

10.1. Трюки с HAL

Основной файл конфигурации X.Org называется /usr/local/etc/X11/xorg.conf. Но в современных версиях X.Org данный файл вообще отсутствует, поскольку вся настройка осуществляется через HAL[1] (вот для этого мы и запускали сервис hald). А это означает, что при простых конфигурациях, которые удовлетворят 90% пользователей, настраивать вообще ничего не придется. Все сделает за вас HAL — никаких настроек (все в автоматическом режиме), никакого конфигурационного файла.

Однако, если вы не входите в эти 90%, настройка гораздо усложняется, поскольку вам придется составлять правила HAL, которые обычному человеку плохо понятны, да и плохо документированы.

10.1.1. Включение <Ctrl>+<Alt>+<Backspace>

Комбинация клавиш <Ctrl>+<Alt>+<Backspace> используется для аварийного завершения X.Org в случае ее зависания. Никогда не знаешь, когда пригодится эта замечательная комбинация. Раньше она была активной по умолчанию, в последних версиях X.Org ее нужно включить вручную. А сделать это можно, как вы уже догадались, через HAL.

Файлы политик (правил) HAL представляют собой XML-файлы. Сейчас мы отредактируем файл /usr/local/etc/hal/fdi/policy/x11-input.fdi, влияющий на ввод с клавиатуры.

---

[1] HAL (Hardware Abstraction Layer (Слой аппаратных абстракций)) — реализованный в программном обеспечении слой абстрагирования, находящийся между физическим уровнем аппаратного обеспечения и программным обеспечением, запускаемым на данном компьютере. HAL предназначен для скрытия различий в аппаратном обеспечении от основной части ядра операционной системы таким образом, чтобы большая часть кода, работающая в режиме ядра, не нуждалась в изменении при ее запуске на системах с различным аппаратным обеспечением. На персональных компьютерах HAL, по существу, может рассматриваться как драйвер материнской платы, позволяющий взаимодействовать инструкциям высокоуровневых языков программирования с низкоуровневыми компонентами, такими как аппаратное обеспечение. — *Из Википедии*.

Создайте файл /usr/local/etc/hal/fdi/policy/x11-input.fdi:

```
# ee /usr/local/etc/hal/fdi/policy/x11-input.fdi
```

В него нужно добавить следующие строки:

```
<?xml version="1.0" encoding="UTF-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="info.capabilities" contains="input.keyboard">
      <merge key="input.x11_options.XkbOptions"
type="string">terminate:ctrl_alt_bksp</merge>
    </match>
  </device>
</deviceinfo>
```

Сохраните файл и перезапустите сервис HAL:

```
# /usr/local/etc/rc.d/hald restart
```

10.1.2. Запрет опроса устройств

Демон HAL по умолчанию опрашивает устройства при их подключении к компьютеру. Например, он может опросить USB-контроллер, когда вы подключите флешку, внешний жесткий диск или любое другое USB-устройство. Иногда такой безобидный опрос устройства заканчивается зависанием системы. Наиболее часто подобная проблема возникает именно с USB-устройствами. Поэтому отключим опрос устройств для контроллера uhci0 (обычно это первый USB-контроллер). К этому контроллеру и нужно будет подключать все проблемные устройства.

Перейдите в каталог /usr/local/etc/hal/fdi/preprobe. Создайте каталог 20thirdparty, в нем — файл ignore-uhci0.fdi:

```
# cd /usr/local/etc/hal/fdi/preprobe
# mkdir 20thirdparty
# ee ignore-uhci0.fdi
```

Содержимое файла ignore-uhci0.fdi должно быть таким:

```
<?xml version="1.0" encoding="UTF-8"?>
<deviceinfo version="0.2">
  <device>
    <match key="freebsd.driver" string="uhci">
      <match key="freebsd.unit" int="0">
        <merge key="info.ignore" type="bool">true</merge>
      </match>
    </match>
  </device>
</deviceinfo>
```

В каталоге /usr/local/share/hal/fdi/preprobe/10osvendor вы найдете еще два примера fdi-файлов:

- 10-ide-drives.fdi — обработка некоторых IDE-дисков;
- 20-broken-usb-sticks.fdi — обработка некорректно работающих флешек.

10.1.3. Монтирование устройств с помощью HAL

В Linux при подключении сменного носители (флешки, привода CD/DVD) в GNOME на рабочем столе сразу же появляется значок этого носителя, то есть монтирование осуществляется автоматически, что очень удобно. В FreeBSD такого нет, и мы сейчас это исправим.

Первым делом нужно отметить, что носители, которые вы хотите монтировать автоматически, не должны быть указаны в /etc/fstab. Особенно это касается флешек и приводов CD/DVD. После установки системы в /etc/fstab автоматически добавляется строка монтирования привода CD/DVD. Ее нужно удалить или закомментировать.

Затем следует подключить псевдофайловую систему procfs *(см. разд. 9.3)*.

Чтобы монтировать устройства с помощью HAL, вы должны быть авторизированным пользователем. Если вы запускаете GNOME через GDM *(см. разд. 9.3)*, то авторизацией занимается непосредственно сам GDM, и вам не нужно больше выполнять никаких настроек. А вот если вы вызываете GNOME вручную, то следует отредактировать файл /usr/local/etc/PolicyKit/PolicyKit.conf. В него надо добавить действие `org.freedesktop.hal.storage.mount-removable` для вашего пользователя:

```
<match action="org.freedesktop.hal.storage.mount-removable">
  <match user="denis">
    <return result="yes"/>
  </match>
</match>
```

Естественно, что вместо `"denis"` нужно указать имя вашего пользователя.

Если на диске имеются разделы других операционных систем, добавьте в этот файл строки, разрешающие пользователю `denis` монтировать стационарные носители:

```
<match action="org.freedesktop.hal.storage.mount-fixed">
  <match user="denis">
    <return result="yes"/>
  </match>
</match>
```

Сама операция монтирования в FreeBSD недоступна обычным пользователям. Поэтому вне зависимости от того, как вы запускаете GNOME, вам нужно разрешить обычным пользователям монтировать носители данных. Для этого введите команду:

```
# sysctl vfs.usermount=1
```

После этого всех пользователей, которым вы намерены разрешить монтирование, поместите в группу `operator`:

```
# pw group mod operator -m denis
```

Если вы хотите, чтобы Nautilus монтировал другие разделы (например, Windows-разделы), указанные в файле /etc/fstab, укажите для этих разделов опцию `noauto`:

```
/dev/ad0s1          /mnt/win_c          msdosfs rw,noauto       0 0
```

Точка монтирования /`mnt/win_c` должна существовать и должна принадлежать пользователю, который будет осуществлять монтирование:

```
# mkdir /mnt/win_c
# chown denis /mnt/win_c
```

10.2. Редактор конфигурации gconf-editor

Конфигурацию GNOME можно изменить с помощью редактора конфигурации gconf-editor. Особенно приятно, что в FreeBSD эта программа доступна сразу (в Linux приходится доустанавливать пакет gconf-editor). Редактор gconf-editor можно сравнить с редактором реестра Windows, но gconf-editor изменяет настройки не всей системы, а только GNOME.

Начиная с версии GNOME 2.28, у некоторых пунктов меню отсутствуют пиктограммы (рис. 10.1). Посмотрите: в меню **Система** пиктограмм нет, а в меню **Система | Параметры** — есть. Меню GNOME будет смотреться гораздо лучше, если отсутсвующие пиктограммы добавить. Для этого нажмите комбинацию клавиш <Alt>+<F2> и введите команду `gconf-editor`.

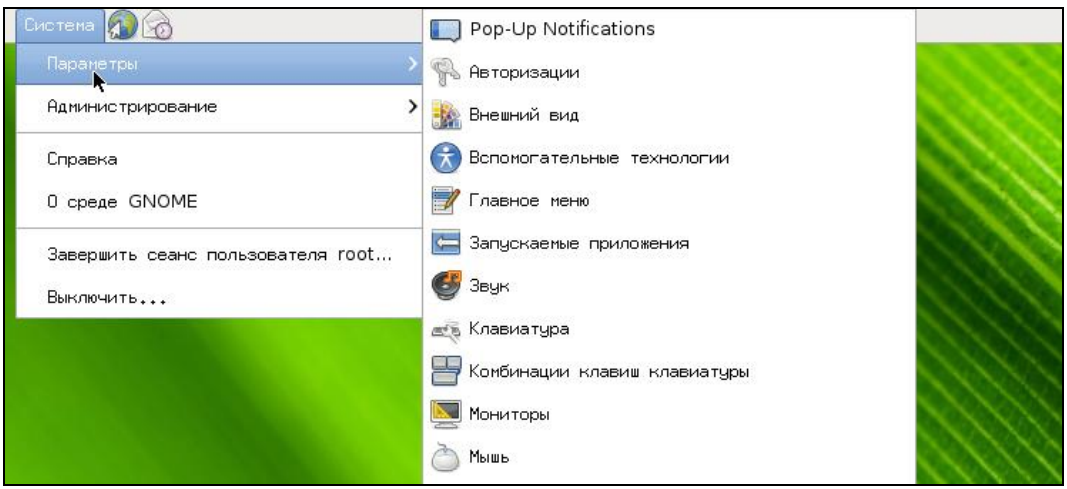


Рис. 10.1. Меню GNOME

Когда запустится редактор конфигурации, перейдите в раздел **/desktop/gnome/interface** и включите параметр **menus_have_icons** (рис. 10.2). Попробуйте теперь открыть меню **Система** (перезагружать GNOME не нужно — изменения вступают в силу мгновенно). Стало намного лучше, да (рис. 10.3)?

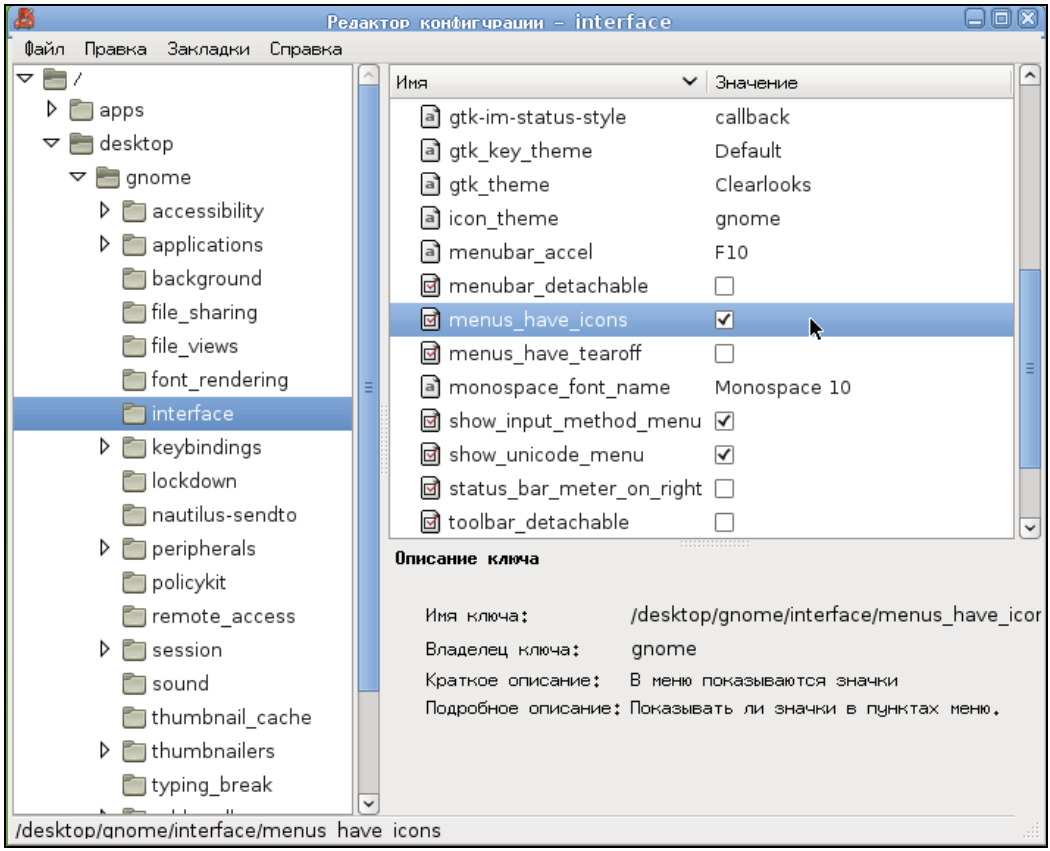


Рис. 10.2. Редактор конфигурации gconf-editor

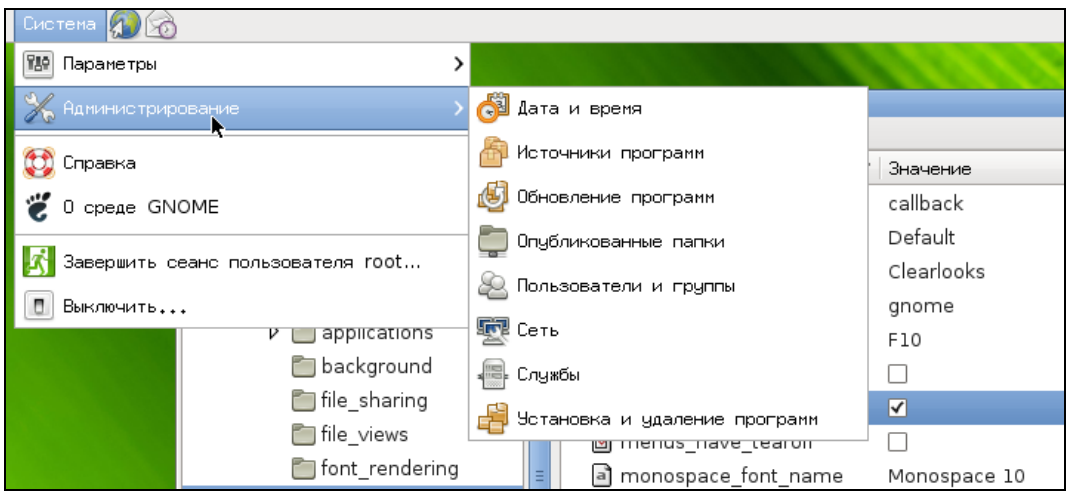


Рис. 10.3. Меню после проделанных изменений

Но это еще не все. Вы можете определить, какие значки должны быть на рабочем столе. По умолчанию на него вынесены значки **Компьютер**, **Домашняя папка пользователя**, **Корзина** и значки носителей.

Снова откройте редактор конфигурации и перейдите в раздел **/apps/nautilus/desktop** (рис. 10.4). Включите (или выключите) следующие параметры:

- **computer_icon_visible** — управляет отображением пиктограммы **Компьютер** (параметр **computer_icon_name** задает имя пиктограммы);
- **home_icon_visible** — управляет отображением пиктограммы **Домашняя папка пользователя** (параметр **home_icon_name** задает имя пиктограммы);
- **network_icon_visible** — управляет отображением пиктограммы **Сеть** (параметр **network_icon_name** задает имя пиктограммы);
- **trash_icon_visible** — управляет отображением пиктограммы **Корзина** (параметр **trash_icon_name** задает имя пиктограммы);
- **volumes_visible** — определяет, будут ли показаны на рабочем столе подключенные тома (пиктограммы сменных носителей и разделов жесткого диска).

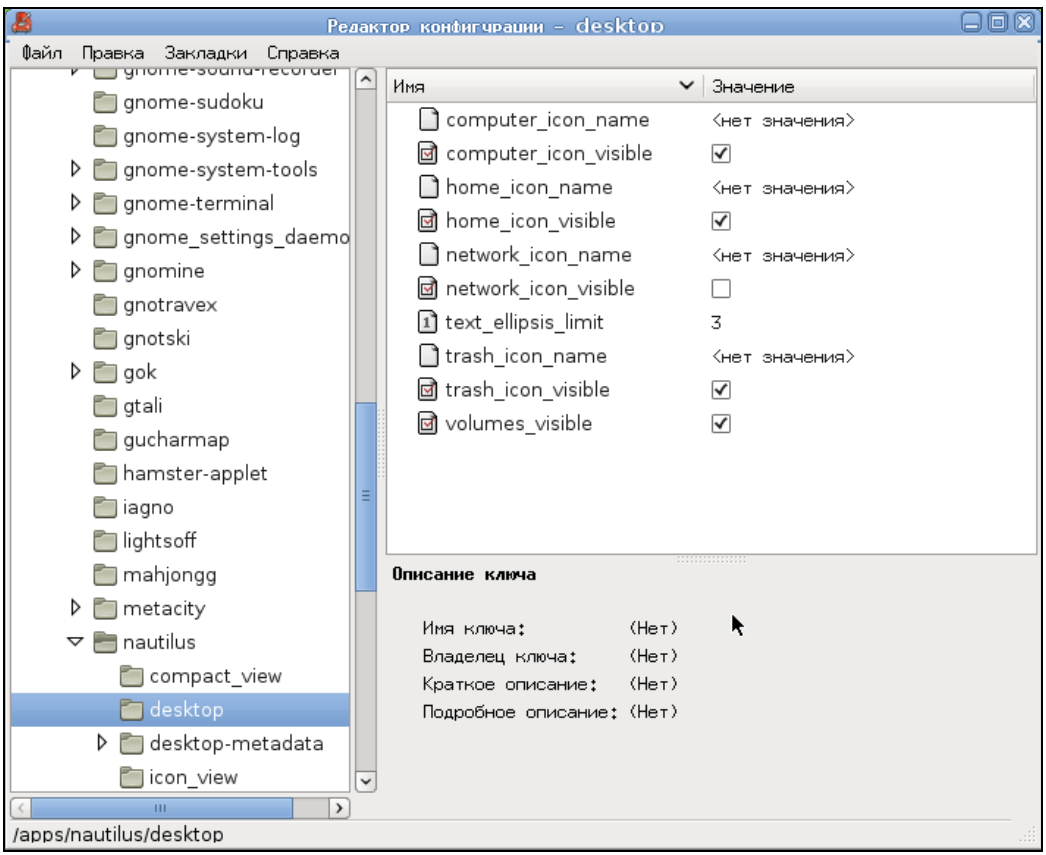


Рис. 10.4. Раздел **/apps/nautilus/desktop** редактора конфигурации

Рассмотрим еще некоторые интересные параметры GNOME, находящиеся в разделе /**desktop/gnome**:

- в разделах **/desktop/gnome/accessibility/keyboard** и **/desktop/gnome/accessibility/mouse** каждый параметр тщательно документирован (тем более, описание их приводится на русском языке), поэтому вы разберетесь с ними и без моих комментариев;
- в разделе **/desktop/gnome/applications** находятся параметры некоторых GNOME-приложений. Так, раздел **/desktop/gnome/applications/window_manager** содержит параметры диспетчера окон. Параметр **number_of_workspaces** задает количество рабочих столов GNOME;
- в разделе **/desktop/gnome/background** находятся параметры фона рабочего стола. Если выключить параметр **draw_background**, вы, по сути, отключите рабочий стол. Параметр **picture_filename** задает имя файла, использующегося в качестве обоев рабочего стола. Параметр **picture_options** задает параметры картинки. По умолчанию картинка растягивается на весь рабочий стол (значение **stretched**), но вы можете установить значение **center**, чтобы картинка отображалась по центру;
- в разделе **file_sharing** находятся параметры общего доступа к файлам, в том числе и по Bluetooth. Включить Bluetooth можно, активизировав параметр **bluetooth_enable**. Непосредственно сам общий доступ к файлам включается параметром **enabled**;
- раздел **file_views** содержит параметры отображения файлов. Например, параметр **show_hidden_files** позволяет включить отображение скрытых файлов;
- некоторые опции раздела **interface** уже были рассмотрены ранее. Просмотрите остальные параметры — вы найдете много чего интересного;
- в разделе **lockdown** вы найдете параметры блокировки:
  - **disable_command_line** — будет отключен доступ к терминалу и к командной строке. Окно **Выполнить программу** (при нажатии комбинации <Alt>+<F2>) тоже будет отключено;
  - **disable_lock_screen** — запрещает пользователю блокировать рабочий стол;
  - **disable_printing** — запрещает печать;
  - **disable_print_setup** — запрещает настройку печати;
  - **disable_save_to_disk** — запрещает опцию **Сохранить как** во всех приложениях;
  - **disable_user_switching** — запрещает переключение пользователей;
- параметры удаленного доступа находятся в разделе **remote_access;**
- отключить звуки событий можно в разделе **sounds**, выключив параметр **event_sounds**;
- когда файловый менеджер GNOME (программа Nautilus) отображает содержимое каталога, он пытается создать миниатюры для каждого типа файла (для видеофайлов, картинок и т. д.). Для создания миниатюры вызывается сторонняя программа, а какая именно, "прописано" в разделе **thumbnailers**. В нем вы найдете

множество подразделов — по одному для каждого типа файла. Соответственно, в каждом подразделе указывается, какая именно программа будет использоваться для создания миниатюры для файла того или иного типа;

- аналогично, в разделе **url_handlers** указываются обработчики URL для каждого типа URL (HTTP, HTTPS, FTP, MAIL и т. п.);
- в разделе **/apps/nautilus/preferences** содержится интересный параметр **desktop_is_home_dir**. Если его включить, то GNOME будет использовать домашнюю папку пользователя в качестве рабочего стола. Другими словами, все, что есть в вашем домашнем каталоге, будет отображено на рабочем столе. Кому-то такое поведение системы понравится, кому-то нет, — на любителя.

10.3. Поддержка видеокарт NVIDIA

Для поддержки видеокарт NVIDIA установите порт x11/nvidia-driver:

```
# cd /usr/ports/x11/nvidia-driver
# make install clean
```

А затем обеспечьте автоматическую загрузку модуля `nvidia`:

```
# echo kldload nvidia >> /etc/rc.local
```



ЧАСТЬ III

Командная строка



Глава 11

Выбор и использование командной оболочки

11.1. Файл /etc/shells

Для обычных пользователей в FreeBSD по умолчанию предназначен командный интерпретатор (так называемая *оболочка*) sh. Для суперпользователя root по умолчанию используется оболочка csh, точнее, tcsh.

Основное назначение оболочки — это выполнение команд, введенных пользователем. Пользователь вводит команду, оболочка проверяет, является ли эта команда внутренней командой или это внешняя программа. Если введенная команда должна запустить внешнюю программу, оболочка ищет программу, соответствующую команде, в каталогах, указанных в переменной окружения PATH. Если такая программа найдена, то оболочка запускает ее и передает ей введенные пользователем параметры. В противном случае выводится сообщение о невозможности выполнения команды.

Кроме sh и tcsh существуют и другие оболочки: bash, csh, ksh, zsh, ash и пр. Все командные оболочки, установленные в системе, прописаны в файле /etc/shells, и количество их может быть довольно велико. В листинге 11.1 приведен список оболочек, установленных в моей системе.

Необходимо отметить, что оболочка bash по умолчанию не устанавливается. Если вы раньше работали с Linux, то, наверняка, привыкли к bash. Для ее установки нужно ввести команду:

```
# pkg_add -r bash
```

Ну, а оболочка rbash устанавливается одновременно с bash.

Листинг 11.1. Файл /etc/shells

```
# $FreeBSD: src/etc/shells,v 1.5.36.1.4.1 2010/06/14 02:09:06 kensmith Exp $
#
# List of acceptable shells for chpass(1).
# Ftpd will not allow users to connect who are not using
# one of these shells.

/bin/sh
/bin/csh
```

```
/bin/tcsh
/usr/local/bin/bash
/usr/local/bin/rbash
```

С точки зрения пользователя указанные оболочки мало чем друг от друга отличаются. И все они позволяют выполнять введенные пользователем команды. Но оболочки используются не только для выполнения команд, а еще и для автоматизации задач с помощью *сценариев*. Так вот, все эти оболочки отличаются синтаксисом языка описания сценариев.

11.2. Разнообразие выбора

11.2.1. Оболочка sh

Самым первым командным интерпретатором (оболочкой) в операционной системе UNIX стала sh (сокращение от shell). Оболочку sh разработал Стивен Борн (Steven Bourne), поэтому ее второе название Оболочка Борна (Bourne Shell). Изначально sh была создана для операционной системы AT&T (разработка Bell Labs). Чуть позже sh усовершенствовали, и она вошла в состав POSIX (Portable Operating System Interface for Unix, переносимый интерфейс операционных систем UNIX). Усовершенствованная версия sh до сих пор устанавливается в современных версиях FreeBSD (вот только зачем?).

С точки зрения пользователя оболочка sh не очень удобна, поэтому пользователи предпочитают ей другую оболочку, например, tcsh или bash. Не побоюсь быть категоричным — по сравнению с bash или tcsh, оболочка sh — просто убожество. Не понимаю почему, но в FreeBSD она устанавливается по умолчанию для обычных пользователей. Хорошо, что для пользователя root по умолчанию используется tcsh.

11.2.2. Оболочка csh

Оболочка csh (C Shell) по умолчанию устанавливается в FreeBSD. Разработка csh началась еще с первых версий BSD. Тогда в институте Беркли начали создавать эту новую оболочку (csh), потому что не захотели мириться с ограничениями sh.

Внутренний синтаксис csh очень напоминает язык программирования C, поэтому он должен был понравиться программистам (а в то время все пользователи компьютеров являлись программистами). Тем не менее сами программисты отмечали, что синтаксис csh не очень удобен, даже несмотря на то, что он похож на C.

По сравнению с первой версией sh у оболочки csh было множество преимуществ: она умела управлять заданиями, хранить историю ранее введенных команд, а также включала сценарии, выполняемые при входе пользователя (запуске оболочки) и при его выходе (когда пользователь вводит команду `exit`), что было очень удобно (в то время sh не имела таких сценариев). С точки зрения обычного использования оболочки (а не программирования) csh тоже была на высоте.

В последних версиях FreeBSD вместо csh используется ее усовершенствованная версия tcsh, а файл /bin/csh — это просто ссылка на файл /bin/tcsh. Даже при вводе команды `man csh` открывается страница руководства tcsh.

11.2.3. Оболочка ksh

Не хочется делать экскурс в историю UNIX, но пару слов сказать все же придется. Изначально UNIX появилась в лабораториях компании AT&T, позже появились версии UNIX от института Беркли (операционная система BSD). Так уж сложилось исторически, что AT&T и институт Беркли постоянно конкурировали между собой. Как только в Беркли создали оболочку csh, в AT&T принялись разрабатывать собственную оболочку, которая получила название ksh (Korn Shell) — по имени разработчика Девида Корна (David Korn).

Оболочка ksh по функциям похожа на csh: есть поддержка управления заданиями, история команд, она позволяет назначать командам псевдонимы, а также создавать конфигурационные файлы для подоболочек.

Несмотря на то, что оболочка была разработана в 1986 году, она до сих пор используется в некоторых версиях UNIX по умолчанию. Правда, изначально ksh — это коммерческий продукт, поэтому в FreeBSD используется не ksh, а ее бесплатная версия — pdksh. Впрочем, для краткости исполнимый файл оболочки все равно называется ksh.

Начинающим пользователям ksh не понравится (лучше использовать bash) — она слишком неудобна в использовании, зато у нее довольно развитый синтаксис внутреннего языка, что нравится программистам.

11.2.4. Оболочка bash

Командный интерпретатор bash (Bourne Again Shell, Еще одна разработка Борна) создан фондом свободного программного обеспечения (Free Software Foundation, FSF). За основу была взята оболочка sh.

Оболочка bash может применяться также и для запуска сценариев sh. Поэтому установив в FreeBSD оболочку bash, вы по-прежнему сможете выполнять sh-сценарии.

С точки зрения пользователей bash намного удобнее, чем ksh. Вы можете легко редактировать командную строку, просматривать историю команд, создавать псевдонимы команд, создавать переменные окружения и использовать их в собственных сценариях. Как и в csh, в bash имеются сценарии, которые вызываются при запуске оболочки и при выходе из нее.

Синтаксис bash довольно прост, поэтому большая часть сценариев, разрабатываемых в Linux, пишутся именно на bash. Разработчики FreeBSD обычно более консервативны — они для написания сценариев выбирают sh, впрочем, bash во многом совместима с sh, так что можно особо на этот счет не беспокоиться. Зато bash намного удобнее в использовании, чем sh. Поэтому я рекомендую сменить sh на bash. И чем скорее вы это сделаете, тем лучше. Для смены оболочки воспользуйтесь командой:

```
chsh -s /usr/local/bin/bash <имя пользователя>
```

Далее мы будем предполагать, что у вас установлена именно оболочка bash. Конечно, я расскажу и об особенностях оболочки sh — для тех читателей, которым лень ввести команду `chsh`.

Примечание

В *разд. 11.3* и *12.1* оболочка bash будет рассмотрена более подробно.

11.2.5. Оболочка zsh

Оболочки bash и tcsh (современная версия csh) будут подробно рассмотрены в *разд. 11.3* и *11.4* соответственно, а также в *главе 12*, оболочка ksh используется редко. Поэтому сейчас мы поближе познакомимся с оболочкой zsh, которая становится все более популярной.

До того как познакомиться с zsh, я считал оболочку bash самой удобной. Однако это оказалось не так.

Что же удобного в zsh. Во-первых, это навигация. В bash для перехода в каталог /dir/subdir1/subdir2 нужно ввести команду:

```
cd /dir/subdir1/subdir2
```

Можно использовать автодополнение bash и вводить начальные символы каталога, нажимая каждый раз клавишу <Tab>. Это будет выглядеть примерно так:

```
cd /dir/sub <Tab>/subdi <Tab>
```

В zsh можно ввести: `/d/s/s` и нажать клавишу <Tab> — вы перейдете в нужный каталог. Для перехода в каталог /etc/sysconfig/network в zsh нужно ввести `/e/s/n` и нажать клавишу <Tab>. Обратите, кстати, внимание — команда `cd` уже не нужна.

Покажу еще один трюк. Предположим, у нас есть каталог files, а в нем вложенные каталоги f1 и f2. Внутри каждого каталога f* имеются каталоги source и last. То есть структура каталогов выглядит примерно так:

```
/files/f1/sources/last
/files/f2/sources/last
```

Пусть мы находимся в каталоге /files/f1/sources/last, тогда для перехода в каталог /files/f2/sources/last в zsh достаточно ввести команду: `cd 1 2`.

Но одной лишь навигацией возможности zsh не ограничиваются. Можно, например, использовать вот такое перенаправление:

```
< /var/log/messages
```

Оболочка запустит программу, указанную в переменной `$PAGER`. В большинстве случаев это аналогично команде:

```
cat /var/log/messages | less
```

Все возможности zsh здесь мы рассматривать не будем — их весьма много. Если вы заинтересовались, обратитесь к следующим страничкам:

- **http://opennet.ru/base/dev/zsh_intro.txt.html**;
- **http://citkit.ru/articles/1083/**;
- **http://alexott.net/ru/writings/zsh/index.html**;
- **http://habrahabr.ru/blogs/linux/82537/**.

11.2.6. Оболочка tcsh

Оболочка tcsh является модифицированной версией csh. Буква t в названии означает TENEX — изначально оболочка была разработана для операционной системы TENEX (использовалась в далеком прошлом на компьютерах DEC PDP-10).

В tcsh усовершенствована функция редактирования командной строки, включено автозавершение команд (как в bash). Кроме того, tcsh может распознавать потенциально опасные команды. Если вы от имени root попытаетесь удалить все файлы, оболочка потребует подтверждения.

Оболочка tcsh очень удобна в использовании, но ее синтаксис сценариев сложнее, чем bash. Однако в *главе 12* мы все же рассмотрим разработку сценариев на tcsh, чтобы вы смогли сравнить сложность создания сценариев на bash и на tcsh.

Напомню, что в FreeBSD данная оболочка используется по умолчанию для пользователя root.

Примечание

В *разд. 11.4* и *12.2* оболочка tcsh будет рассмотрена более подробно.

11.2.7. Оболочка ash

Almquist shell (ash) — самая простая и самая маленькая командная оболочка, доступная для UNIX, у нее и самые низкие требования к дисковому пространству.

У ash всего 24 встроенные команды и 10 опций командной строки. Именно поэтому она такая компактная. Зато ее возможности оставляют желать лучшего — вряд ли вы будете использовать ash для повседневной работы.

11.2.8. Выбор оболочки

Какую оболочку выбрать? Первым делом нужно оценить простоту использования оболочки. Ведь вы будете работать с ней каждый день, поэтому простота использования должна быть на первом месте.

Затем оцените простоту синтаксиса оболочки. Конечно, это только в том случае, если вы планируете разрабатывать собственные сценарии. Также не нужно забывать, что вы можете использовать одну оболочку, а разрабатывать сценарии — на языке другой оболочки.

Лично я в своей системе установил оболочку bash по умолчанию для всех пользователей, в том числе и для пользователя root. Мне удобно использовать эту оболочку каждый день и мне удобно разрабатывать сценарии на языке этой оболочки. Вы же можете поэкспериментировать с tcsh. В повседневном использовании — довольно удобная оболочка. Но для написания сценариев я бы выбрал bash.

11.3. Оболочка bash

bash — это наиболее часто использующаяся командная оболочка (командный интерпретатор) Linux. В FreeBSD bash устанавливается по желанию пользователя.

Общее понятие о работе в консоли мы получили из *разд. 8.2*, поэтому сразу перейдем к рассмотрению конфигурационных файлов оболочек bash и sh.

Конфигурационный файл /etc/profile содержит глобальные настройки оболочек sh и bash, они влияют на всю систему — на каждую запущенную оболочку. Обычно файл /etc/profile не нуждается в изменении, а при необходимости изменить параметры sh/bash редактируют один из следующих файлов (напомню, что символом ~ обозначается домашний каталог пользователя):

- ~/.profile и ~/.bash_profile — обрабатываются при каждом входе в систему (sh читает только первый файл, bash читает оба файла);
- ~/.shrc и ~/.bashrc — обрабатывается при каждом запуске дочерней оболочки (первый файл читается оболочкой sh, второй — оболочкой bash);
- ~/.bash_login — читается при входе пользователя в систему (при условии, что bash является оболочкой по умолчанию);
- ~/.bash_logout — обрабатывается при выходе из системы (только оболочкой bash).

> ***Примечание***
>
> Оболочка bash читает файлы .bash_profile, .profile и .bash_login только, если она является оболочкой по умолчанию, так называемой *login-оболочкой* (login-shell). Если вы вызвали bash из другой оболочки, скажем, из sh, то будет прочитан только файл .bashrc.

Файл ~/.bash_profile часто не существует, а если и существует, то в нем есть всего одна строка:

```
source ~/.bashrc
```

Отсюда следует, что основным конфигурационным файлом оболочки bash служит файл .bashrc. Но помните, что он влияет на оболочку текущего пользователя (такой файл находится в домашнем каталоге каждого пользователя, не забываем также, что символ ~ означает домашний каталог). Если же вдруг понадобится задать параметры, которые повлияют на всех пользователей, придется редактировать файл /etc/profile.

В файле .bash_history (он тоже находится в домашнем каталоге) хранится история команд, введенных пользователем. Там вы можете просмотреть свои же команды, которые накануне вводили.

Какие настройки могут быть занесены в файл в .bashrc (.shrc)? Как правило, в этом файле задаются псевдонимы команд, определяется внешний вид приглашения командной строки, задаются значения переменных окружения. Синтаксис обоих файлов (.bashrc и .shrc) одинаков, просто они обрабатываются разными оболочками.

Псевдонимы команд задаются с помощью команды `alias`, вот несколько примеров:

```
alias h='fc -l'
alias ll='ls -laFo'
alias l='ls -l'
alias g='egrep -i'
alias df='df -h'
```

Псевдонимы работают просто — так, настройка `alias l='ls -l'` означает, что при вводе команды `l` на самом деле будет выполнена команда `ls -l`.

Теперь рассмотрим пример изменения приглашения командной строки. Глобальная переменная `PS1` отвечает за внешний вид командной строки. По умолчанию приглашение командной строки bash имеет формат:

```
пользователь@компьютер:рабочий_каталог
```

Значение `PS1` при этом будет:

```
export PS1='\u@\h:\w$'
```

В табл. 11.1 приведены допустимые модификаторы командной строки.

Таблица 11.1. *Модификаторы командной строки*

Модификатор	Описание
`\a`	ASCII-символ звонка (код 07). Не рекомендуется его использовать — очень скоро начнет раздражать
`\d`	Дата в формате "День недели, Месяц, Число"
`\h`	Имя компьютера до первой точки
`\H`	Полное имя компьютера
`\j`	Количество задач, запущенных в оболочке в данное время
`\l`	Название терминала
`\n`	Символ новой строки
`\r`	Возврат каретки
`\s`	Название оболочки
`\t`	Время в 24-часовом формате (ЧЧ: ММ: СС)
`\T`	Время в 12-часовом формате (ЧЧ: ММ: СС)
`\@`	Время в 12-часовом формате (AM/PM)
`\u`	Имя пользователя
`\v`	Версия bach (сокращенный вариант)
`\V`	Версия bash (полная версия: номер релиза, номер патча)
`\w`	Текущий каталог (полный путь)
`\W`	Текущий каталог (только название каталога, без пути)
`\!`	Номер команды в истории
`\#`	Системный номер команды
`\$`	Если UID пользователя равен 0, будет выведен символ #, иначе — символ $
`\\`	Обратный слэш
`$ ( )`	Подстановка внешней команды

Вот пример задания альтернативного приглашения командной строки:

```
export PS1='[\u@\h] $(date +%m/%d/%y) \$'
```

Вид приглашения будет такой:

[denis@host] 12/06/10 $

В квадратных скобках выводится имя пользователя и имя компьютера, затем используется конструкция **$()** для подстановки даты в нужном нам формате и символ приглашения, который изменяется в зависимости от идентификатора пользователя.

11.4. Оболочка tcsh

Основные интерактивные возможности tcsh следующие:

- редактирование командной строки;
- дополнение слов — как команд, так и путей;
- хранение и возможность просмотра истории команд;
- управление задачами.

Редактирование командной строки осуществляется с помощью клавиш <←> и <→> (перемещение курсора по командной строке) и клавиш <Delete> и <Backspase> (удаление символов). Как видите, все просто.

Для автоматического дополнения слов может, как и в bash, использоваться клавиша <Tab>, но правильнее использовать комбинации клавиш <Ctrl>+<I> и <Ctrl>+<D>. Первая обеспечивает автодополнение слова, а вторая выводит список подходящих для автодополнения вариантов.

Просмотреть историю ранее введенных команд можно с помощью клавиш <Up> и <Down>. Если же эти клавиши почему-то не работают (меньше нужно играть в NFS!), то можно использовать команду `history`:

```
$ history
1       13:08   clear
2       13:09   builtins
3       13:19   history
```

Вызвать любую команду из истории можно с помощью конструкции `!#`, где `#` — это номер команды, например: `!1`

Управление задачами осуществляется так же, как и в bash. Чтобы запустить команду в фоновом режиме, нужно добавить к ней символ `&`, например: `$ command &`

Вывести список запущенных в фоновом режиме задач можно командой `jobs`. Команда `fg` переводит задачу в активный режим (foreground), нужно указать ее номер (полученный командой `jobs`), например: `$ fg 1`.

Команда `bg` переводит активную задачу в фоновый режим (background). Для нее, как и для команды `fg`, нужно указать номер задачи: `$ bg 1`.

Просмотреть список встроенных в оболочку команд можно командой `builtins` (рис. 11.1). У tcsh насчитывается более 70 встроенных команд. В табл. 11.2 представлены самые полезные из них.

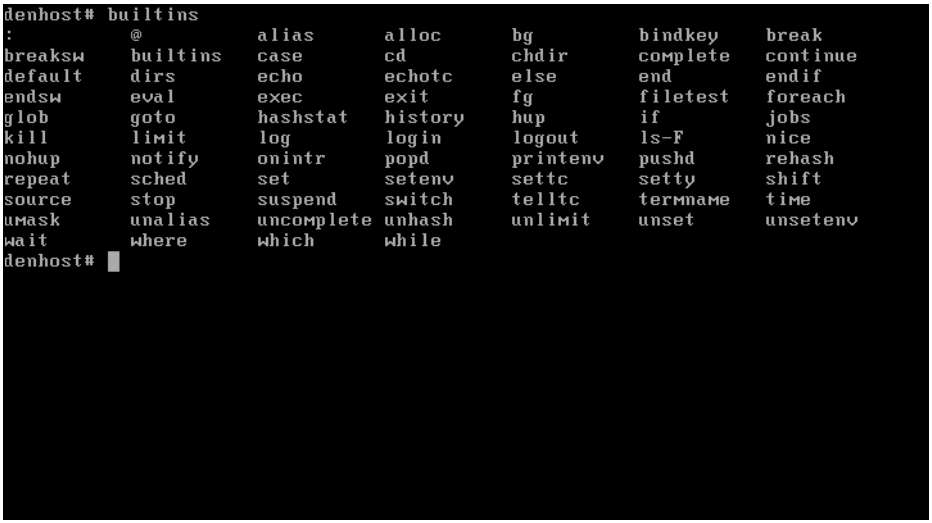


Рис. 11.1. Встроенные команды оболочек tcsh

Таблица 11.2. *Самые полезные встроенные команды tcsh*

Команда	Описание
`@`	Вычисляет арифметическое выражение. Данная команда будет подробно описана при написании сценариев *(см. разд. 12.2.1)*
`alias`	Создает псевдоним для команд оболочки, эту команду мы рассмотрели ранее
`alloc`	Отображает отчет о количестве свободной и использованной памяти
`bg`	Перемещает приостановленную задачу в фоновый режим
`builtins`	Отображает список встроенных команд
`cd` или `chdir`	Изменяет текущий каталог
`dirs`	Отображает стек каталогов
`echo`	Отображает свои аргументы. Обычно используется при написании сценариев
`exec`	Запускает другую программу в этой же оболочке
`exit`	Осуществляет выход из tcsh
`fg`	Перемещает задачу на "передний план", то есть делает ее активной
`filetest`	В основном используется при написании сценариев. Позволяет проверить тип файла (устройство, каталог, файл), существование файла и осуществить ряд других полезных проверок
`glob`	Похожа на `echo`, но не отображает пробелы между аргументами и не выводит символ новой строки после своего вывода

Таблица 11.2 *(продолжение)*

Команда	Описание
`hashstat`	Выводит статистику механизма хэширования tcsh
`history`	Отображает введенные ранее команды
`jobs`	Отображает список задач (приостановленные команды и те, которые выполняются в фоновом режиме)
`kill`	Завершает задачу или процесс
`limit`	Позволяет ограничить ресурсы текущего процесса и всех процессов, которые он будет создавать
`login`	Используется для входа пользователя. Сопровождается именем пользователя
`logout`	Завершает сессию, если вы используете tcsh в качестве оболочки по умолчанию
`ls-F`	Команда `ls-F` аналогична по действию внешней программе `ls`, запускаемой с параметром `-F`, но выполняется быстрее за счет того, что это встроенная команда оболочки. На современных компьютерах вы прироста производительности не заметите, команда `ls-F` будет выполняться так же быстро, как и `ls -F`, но не забываем, в какие времена разрабатывалась система BSD и какие тогда были компьютеры. Если задан параметр `-F` программы `ls`, то для каждого имени каталога следует добавлять суффикс `` `/' ``, для каждого имени FIFO — `` `|' `` и для каждого имени исполняемого файла — `` `*' ``
`nice`	Позволяет изменить приоритет процесса (см. `man nice`)
`nohup`	Позволяет вам выйти из системы без завершения процессов, выполняемых в фоновом режиме
`notify`	Уведомляет вас при изменении статуса одной из ваших задач
`popd`	Удаляет каталог из стека каталогов
`printenv`	Отображает список всех переменных окружения
`pushd`	Изменяет рабочий каталог и помещает новый каталог на вершину стека каталогов
`rehash`	Пересоздает внутренние таблицы, используемые механизмом хэширования. Допустим, вы создали свой сценарий и поместили его в каталог /usr/bin. Оболочка tcsh не увидит этот файл до тех пор, пока вы не обновите внутренние таблицы
`repeat`	Команде нужно передать два аргумента: количество повторений и простую команду (без потоков и списка команд). `repeat` повторяет эту команду указанное количество раз
`sched`	Выполняет команду в указанное время, например: **$ sched 10:00 echo "Уже 10 часов!"**
`set`	Объявляет и инициализирует локальную переменную

Таблица 11.2 *(окончание)*

Команда	Описание
`setenv`	Объявляет и инициализирует переменную окружения
`source`	Запускает сценарий оболочки, указанный в качестве аргумента. Данная команда не порождает новый процесс, а просто обрабатывает сценарий. Данную команду можно использовать как `include` в обычных языках программирования
`stop`	Останавливает задачу или процесс (которая/который выполняется в фоновом режиме)
`suspend`	Приостанавливает текущую оболочку и помещает ее в фоновый режим
`time`	Запускает команду, указанную в качестве параметра. После выполнения команды отображает информацию о ее выполнении (время выполнения)
`umask`	Изменяет маску прав доступа по умолчанию (см. `man umask`)
`unalias`	Удаляет псевдоним команды
`unhash`	Выключает механизм хэширования. См. также `hashstat` и `rehash`
`unlimit`	Удаляет лимиты текущего процесса
`unset`	Удаляет объявление переменной
`unsetenv`	Удаляет объявление переменной окружения
`where`	В качестве аргумента ей нужно передать команду, после чего получите информацию о том, является ли команда встроенной или исполнимым файлом, и где находится этот файл. Похожа на команду `which`
`which`	Подобна стандартной утилите which, но работает быстрее и владеет информацией обо всех командах и псевдонимах. Выводит местонахождение исполнимого файла, если он вообще существует. Поиск производится в каталогах, указанных в переменной окружения `path`

Рассмотрим теперь конфигурационные файлы оболочки tcsh и способы ее настройки.

Основные глобальные файлы конфигурации tcsh: /etc/csh.cshrc, /etc/csh.login, /etc/csh.logout (обратите внимание — имена файлов такие же, как у старой оболочки csh). Первый файл считывается при запуске каждого экземпляра tcsh в интерактивном режиме, второй — только если tcsh является оболочкой по умолчанию для запустившего ее пользователя. Третий файл считывается при выходе из tcsh при условии, что tcsh является оболочкой по умолчанию (login-оболочкой) для этого пользователя.

При регистрации пользователя в домашнем каталоге создаются файлы ~/.cshrc, ~/.login и ~/.logout, которые являются пользовательскими аналогами файлов /etc/csh.cshrc и /etc/csh.login. Содержимое этих файлов копируется из каталога /usr/share/skel/. В пользовательском каталоге также содержится файл ~/.history, содержащий историю введенных команд.

Порядок считывания глобальных и пользовательских конфигурационных файлов задается при компиляции tcsh и может отличаться в разных системах. Обычно сначала считываются глобальные файлы, а затем пользовательские.

В конфигурационных файлах устанавливаются псевдонимы команд (команда `alias`), переменные окружения (команда `setenv`), служебные переменные tcsh (команда `set`), влияющие на поведение оболочки. Определять переменные окружения имеет смысл в файле ~/.login, который считывается только один раз — при входе пользователя в систему, тогда как файл ~/.cshrc перечитывается при запуске каждого экземпляра оболочки.

> ***Пояснение***
>
> Вы ведь можете в одном сеансе запустить несколько оболочек: войдите в систему и выполните команду `tcsh`, затем еще раз `tcsh` — вот уже три экземпляра и запущено: два вы запустили явно, а еще один был запущен при входе в систему. При первом запуске сначала считывается файл .cshrc, а затем — .login.

Рассмотрим пример определения псевдонимов команд:

```
alias hist  history 25
alias rm rm -i
```

Первая команда создает псевдоним `hist`, выводящий последние 25 команд истории. Вторая команда создает псевдоним для команды `rm`, который называется так же — `rm`, а параметр `-i` означает, что при удалении файлов будет сделан запрос.

Вот пример использования команды `set`, которая устанавливает значение переменной `path` (путь поиска программ):

```
set path = (/bin /usr/bin /usr/local/bin /usr/X11R6/bin)
```

Переменные окружения устанавливаются командой `setenv` (в файле ~/.login), например:

```
setenv  EDITOR  nano
```

Здесь мы установили переменную окружения `EDITOR`, задающую текстовый редактор по умолчанию.

> ***На заметку пользователям tcsh***
>
> Если вы установили какую-нибудь программу, а после установки получаете сообщение, что команда запуска этой программы не найдена, введите команду `rehash`, а затем — имя запускаемой программы. Например, я установил командой `pkg_add -r mc` программу mc. После установки попытка ввода команды `mc` успехом не увенчалась — хэш команд не был обновлен. После команды `rehash` я смог запустить mc. Это замечание относится только к оболочке tcsh. В bash такой особенности нет.

11.5. Перенаправление ввода/вывода

Мы рассмотрели возможности разных оболочек. Осталось разобраться только с перенаправлением ввода/вывода. С помощью перенаправления ввода/вывода мы можем перенаправить вывод одной программы в файл или на стандартный ввод другой программы. Например, у вас не получается настроить сеть, и вы хотите

перенаправить вывод команды `ifconfig` в файл, а затем разместить этот файл на форуме, где вам помогут разобраться с проблемой. А можно командой `ps -ax` перенаправить список всех процессов команде `grep`, которая найдет в списке интересующий вас процесс.

Рассмотрим следующую команду:

```
echo "some text" > file.txt
```

Символ > означает, что вывод команды, находящейся слева от этого символа, будет записан в файл, находящийся справа от символа, при этом файл будет перезаписан.

Чуть ранее мы говорили о перенаправлении вывода программы `ifconfig` в файл. Команда будет выглядеть так:

```
ifconfig > ifconfig.txt
```

Если вместо > указано >>, то исходный файл не будет перезаписан, а вывод команды добавится в конец файла:

```
echo "some text" > file.txt
echo "more text" >> file.txt
cat file.txt
```

some text
more text

Кроме символов > и >> для перенаправления ввода/вывода часто употребляется вертикальная черта `|`. Предположим, что мы хотим вывести содержимое файла big_text:

```
cat big_text
```

Но в файле big_text много строк, они быстро проскочат по экрану, и мы ничего не успеем прочитать. Следовательно, целесообразно отправить вывод команды `cat` какой-то программе, которая будет выводить файл на экран постранично, например,

```
cat big_text | more
```

Конечно, этот пример не очень убедительный, потому что для постраничного вывода гораздо удобнее команда `less`:

```
less big_text
```

Вот еще один интересный пример. Допустим, мы хотим удалить файл file.txt без запроса — для этого можно отдать команду:

```
echo y | rm file.txt
```

Команда `rm` запросит подтверждение удаления (нужно нажать клавишу <Y>), но за нас это сделает команда `echo`.

И еще один пример. Пусть имеется большой файл, и нам нужно найти в нем все строки, содержащие подстроку 555-555. Чтобы не делать это вручную, можно воспользоваться командой:

```
cat file.txt | grep "555-555"
```

Надеюсь, приведенная здесь информация сделает вашу работу в командной строке максимально комфортной.



Глава 12

Создание сценариев на языке оболочки

Представим, что нам нужно выполнить резервное копирование всех важных файлов, для чего создать архивы каталогов /etc, /home и /usr. Понятно, что понадобятся три команды вида:

```
tar -cvjf имя_архива.tar.bz2 каталог
```

Затем нам нужно записать все эти три файла на DVD с помощью любой программы для прожига DVD.

Если выполнять данную операцию раз в месяц (или хотя бы раз в неделю), то ничего страшного. Но представьте, что вам нужно делать это каждый день или даже несколько раз в день? Думаю, такая рутинная работа вам быстро надоест. А ведь можно написать *сценарий*, который сам будет создавать резервные копии и записывать их на DVD! Все, что вам нужно, — это вставить чистый DVD перед запуском сценария.

Можно пойти и иным путем. Написать сценарий, который будет делать резервные копии системных каталогов и записывать их на другой раздел жесткого диска. Ведь не секрет, что резервные копии делаются не только на случай сбоя системы, но и для защиты от некорректного изменения данных пользователем. Помню, удалил как-то важную тему форума и ничего не оставалось, как попросить своего хостинг-провайдера сделать откат. Я был приятно удивлен, когда мне предоставили на выбор три резервные копии — осталось лишь выбрать наиболее подходящую. Не думаете же вы, что администраторы провайдера только и занимались тем, что три раза в день копировали домашние каталоги пользователей? Поэтому, автоматизация — штука полезная, и любому администратору нужно знать, как автоматизировать свою рутинную работу.

12.1. Сценарии оболочки bash

12.1.1. Привет, мир!

По традиции напишем первый сценарий, выводящий всем известную фразу: "Привет, мир!" (Hello world!). Для редактирования сценариев вы можете использовать любимый текстовый редактор, например, nano или ее.

Листинг 12.1. Первый сценарий

```
#!/usr/local/bin/bash
echo "Привет, мир!"
```

Первая строка нашего сценария (листинг 12.1) — это указание, что он должен быть обработан программой /usr/local/bin/bash. Обратите внимание — если между `#` и `!` окажется пробел, то данная директива не сработает, поскольку будет воспринята как обычный комментарий. Комментарии начинаются, как вы уже догадались, с решетки:

```
# Комментарий
```

Вторая строка — это оператор `echo`, выводящий нашу строку. Сохраните сценарий под именем hello и введите команду:

```
$ chmod +x hello
```

Для запуска сценария введите команду:

```
./hello
```

На экране вы увидите строку:

Привет, мир!

Чтобы вводить для запуска сценария просто `hello` (без `./`), сценарий нужно скопировать в каталог /usr/bin (точнее, в любой каталог из переменной окружения `PATH`):

```
# cp ./hello /usr/bin
```

12.1.2. Использование переменных в собственных сценариях

В любом серьезном сценарии вы не обойдетесь без использования *переменных*. Переменные можно объявлять в любом месте сценария, но до места их первого применения. Рекомендуется объявлять переменные в самом начале сценария, чтобы потом не искать, где вы объявили ту или иную переменную.

Для объявления переменной используется следующая конструкция:

```
переменная=значение
```

Пример объявления переменной:

```
ADDRESS=www.dkws.org.ua
echo $ADDRESS
```

Обратите внимание на следующие моменты:

- при объявлении переменной знак доллара не ставится, но он обязателен при использовании переменной;
- при объявлении переменной не должно быть пробелов до и после знака `=`.

Значение для переменной указывать вручную не обязательно — его можно прочитать с клавиатуры:

```
read ADDRESS
```

или со стандартного вывода программы:

```
ADDRESS=`hostname`
```

Чтение значения переменной с клавиатуры осуществляется с помощью инструкции `read`. При этом указывать символ доллара не нужно. Вторая команда устанавливает в качестве значения переменной `ADDRESS` вывод команды `hostname`.

В FreeBSD часто используются *переменные окружения*. Это специальные переменные, содержащие служебные данные. Вот примеры некоторых часто используемых переменных окружения:

- `BASH` — полный путь до исполняемого файла командной оболочки bash;
- `BASH_VERSION` — версия bash;
- `HOME` — домашний каталог пользователя, который запустил сценарий;
- `HOSTNAME` — имя компьютера;
- `RANDOM` — случайное число в диапазоне от 0 до 32767;
- `OSTYPE` — тип операционной системы;
- `PWD` — текущий каталог;
- `PS1` — строка приглашения;
- `UID` — ID пользователя, который запустил сценарий;
- `USER` — имя пользователя.

Для установки собственной переменной окружения используется команда `export`:

```
# присваиваем переменной значение
$ADDRESS=ww.dkws.org.ua
# экспортируем переменную – делаем ее переменной окружения
# после этого переменная ADDRESS будет доступна в других сценариях
export $ADDRESS
```

12.1.3. Передача параметров сценарию

Очень часто сценариям нужно передавать различные параметры, например, режим работы или имя файла/каталога. Для передачи параметров используются следующие специальные переменные:

- `$0` — содержит имя сценария;
- `$n` — содержит значение параметра (`n` — номер параметра);
- `$#` — позволяет узнать количество параметров, которые были переданы.

Рассмотрим небольшой пример обработки параметров сценария. Я понимаю, что конструкцию `case-esac` мы еще не рассматривали, но общий принцип должен быть понятен (листинг 12.2).

Листинг 12.2. Пример обработки параметров сценария

```
# сценарий должен вызываться так:
# имя_сценария параметр

# анализируем первый параметр
case "$1" in
  start)
```

```
        # действия при получении параметра start
        echo "Запускаем сетевой сервис"
        ;;
  stop)
        # действия при получении параметра stop
        echo "Останавливаем сетевой сервис"
        ;;
*)
        # действия в остальных случаях
        # выводим подсказку о том, как нужно использовать сценарий и
        # завершаем работу сценария
echo "Usage: $0 {start|stop }"
        exit 1
        ;;
esac
```

Думаю, приведенных комментариев достаточно, поэтому подробно рассматривать работу сценария из листинга 12.2 не будем.

12.1.4. Массивы и bash

Интерпретатор bash позволяет использовать *массивы*. Массивы объявляются подобно переменным. Вот пример объявления массива:

```
ARRAY[0]=1
ARRAY[1]=2

echo $ARRAY[0]
```

12.1.5. Циклы

Как и в любом языке программирования, в bash можно использовать *циклы*. Мы рассмотрим циклы `for` и `while`, хотя вообще в bash доступны также циклы `until` и `select`, но они применяются довольно редко.

Синтаксис цикла `for` выглядит так:

```
for переменная in список
do
команды
done
```

В цикле при каждой итерации переменной будет присвоен очередной элемент списка, над которым будут выполнены указанные команды. Чтобы было понятнее, рассмотрим небольшой пример:

```
for n in 1 2 3;
do
echo $n;
done
```

Обратите внимание — список значений и список команд должны заканчиваться точкой с запятой.

Как и следовало ожидать, наш сценарий выведет на экран следующее:

1
2
3

Синтаксис цикла `while` выглядит немного иначе:

```
while условие
do
команды
done
```

Цикл `while` выполняется до тех пор, пока истинно заданное условие. Подробно об условиях мы поговорим в следующем разделе, а сейчас напишем аналог предыдущего цикла: вывести значения 1, 2 и 3, но с помощью `while`, а не `for`:

```
n=1
while [ $n -lt  4 ]
do
  echo "$n "
  n=$(( $n+1 ));
done
```

12.1.6. Условные операторы

В bash доступно два *условных оператора*, `if` и `case`. Синтаксис оператора `if` следующий:

```
if условие_1 then
       команды_1
elif условие_2 then
       команды_2
...
elif условие_N then
       команды_N
else
       команды_N+1
fi
```

Оператор `if` в bash работает аналогично оператору `if` в других языках программирования. Если истинно первое условие, то выполняется первый список команд, иначе — проверяется второе условие и т. д. Количество блоков `elif`, понятно, не ограничено.

Самая ответственная задача — это правильно составить условие. Условия записываются в квадратных скобках. Вот пример записи условий:

```
# переменная N = 10
[ N==10 ]
# переменная N не равна 10
[ N!=10 ]
```

Операции сравнения указываются не с помощью привычных знаков > или <, а с помощью следующих выражений:

- `-lt` — меньше;
- `-gt` — больше;
- `-le` — меньше или равно;
- `-ge` — больше или равно;
- `-eq` — равно (используется вместо ==).

Применять данные выражения нужно следующим образом:

```
[ переменная выражение значение|переменная ]
```

Например:

```
# N меньше 10
[ $N -lt 10 ]
# N меньше A
[ $N -lt $A ]
```

В квадратных скобках вы также можете задать выражения для проверки существования файла и каталога:

- `-e файл` — условие истинно, если файл существует;
- `-d каталог` — условие истинно, если каталог существует;
- `-x файл` — условие истинно, если файл является исполнимым.

С оператором `case` мы уже немного знакомы, но сейчас рассмотрим его синтаксис подробнее:

```
case переменная in
значение_1) команды_1 ;;
...
значение_N) команды_N ;;
*) команды_по_умолчанию;;
esac
```

Значение указанной переменной по очереди сравнивается с приведенными значениями (`значение_1`, …, `значение_N`). Если есть совпадение, то будут выполнены команды, соответствующие значению. Если совпадений нет, то будут выполнены команды по умолчанию. Пример использования `case` был приведен в листинге 12.2.

12.1.7. Функции

В bash можно использовать функции. Синтаксис объявления функции следующий:

```
имя() { список; }
```

Вот пример объявления и использования функции:

```
list_txt()
{
echo "Выводим текстовые файлы"
ls *.txt
}
```

12.1.8. Примеры сценариев

Сценарий мониторинга журнала

Начнем с простого сценария мониторинга журнала (листинг 12.3). Системные журналы постоянно обновляются, а наш сценарий будет каждые 3 секунды выводить последние 15 строк выбранного вами журнала. Сценарий будет полезен при настройке системы, когда нужно постоянно просматривать журналы, чтобы понять, как система реагирует на новые настройки. Для прекращения работы сценария нужно нажать комбинацию <Ctrl>+<C>.

Листинг 12.3. Мониторинг системного журнала

```
#!/usr/local/bin/bash
# Интервал обновления, в секундах
INT=3

while [ true ]
do
# Выводим последние 15 строк журнала
tail -n 15 $1
# Ждем
sleep $INT
# Две пустые строки
echo; echo
done
```

Формат вызова следующий:

```
./сценарий файл_журнала
```

Например:

```
./script /var/log/messages
```

Переименование файлов

Следующий сценарий сложнее — он ищет в текущем каталоге файлы, в именах которых есть пробелы, и заменяет пробелы на символы подчеркивания (листинг 12.4).

Листинг 12.4. Сценарий rename_blanks

```
#!/usr/local/bin/bash
#

num=0                          # Сколько файлов мы переименовали
```

```
for filename in *           # Перебираем все файлы в текущем каталоге
do
# Передаем имя файла фильтру grep
# Если имя файла содержит пробел, то код завершения
# последней операции равен 0
     echo "$filename" | grep -q " "
     if [ $? -eq 0 ]
     then
# Если код завершения равен 0, переименовываем
# файл
       fname=$filename
       n=`echo $fname | sed -e "s/ /_/g"`
       mv "$fname" "$n"
       let "num += 1"
     fi
done

echo "Переименовано файлов: $num"

exit 0
```

Преобразование систем счисления

Сейчас мы напишем простенький сценарий, преобразующий десятичное число в шестнадцатеричное с помощью программы dc (листинг 12.5). Самим преобразованием будет заниматься программа dc, а наш сценарий только подготовит данные для этой программы.

Листинг 12.5. Сценарий dec_hex

```
#!/usr/local/bin/bash
B=16   # основание системы счисления
# проверяем, является ли параметр $1 числом
if [ -z "$1" ]
then
echo "Использование: dec_hex число"
exit 1
fi
# Передаем число ($1), систему счисления
# программе dc
echo ""$1" "$B" o p" | dc
```

12.2. Сценарии оболочки tcsh

Мы уже знаем, как превратить обычный текстовый файл в сценарий оболочки — нужно в первой строке указать, какую оболочку запустить для обработки сценария:

```
#!/bin/tcsh
```

После этого файл нужно сделать исполнимым (команда `chmod +x`). Все эти тонкости были рассмотрены в *разд. 12.1.1*.

12.2.1. Переменные, массивы и выражения

Переменные, как уже было показано, устанавливаются командой `set`:

```
set name = denis
```

Здесь мы установили переменную `name` (значение `denis`). Вывести значение переменной можно так:

```
echo $name
```

Если вы введете просто команду `set` (без параметров), то увидите список уже установленных переменных, в том числе и служебных (рис. 12.1).

```
denis-desktop:~> set
_       jobs

addsuffix
argv    ()
autoexpand
autolist
csubstnonl
cwd     /home/denix
dirstack        /home/denix
echo_style      both
edit
gid     1000
group   denix
history 100
home    /home/denix
killring        30
owd
path    (/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /sbin /bin /usr/games
)
prompt  %U%m%u:%B%~%b%#
prompt2 %R?
prompt3 CORRECT>%R (y|n|e|a)?
shell   /usr/bin/tcsh
```

Рис. 12.1. Команда `set`

Удалить переменную можно командой `unset`:

```
unset name
```

Тогда при обращении к переменной вы увидите сообщение:

name: Undefined variable

Переменные окружения принято устанавливать командой `setenv` так:

```
setenv переменная значение
```

Обратите внимание, что между именем переменной и значением нет знака равенства:

```
setenv  EDITOR  nano
```

В tcsh также можно создавать массивы. Вот пример создания и использования массива:

```
$ set nums = (one two three four five)
$ echo $nums
one two three four five
$ echo $nums[3]
three
$ echo $nums[1-3]
one two three
```

Как видите, мы можем вывести сразу весь массив, конкретный элемент и диапазон элементов, что очень удобно. Нумерация элементов массива начинается с 1.

Отдельного разговора заслуживают числовые переменные:

- чтобы присвоить переменной число или результат арифметического выражения, нужно использовать символ `@`, при этом символ `$` перед именем переменной указывать не нужно;
- когда вы используете переменные в выражениях, тогда нужно указывать символ `$`, при этом после символа `@` указывать `$` не нужно;
- после символа `@` обязательно должен следовать пробел.

Рассмотрим несколько примеров:

```
$ @ num = 0
$ echo $num
```

0

```
$ @ num = ( 2 + 2 ) * 2
$ echo $num
```

8

```
$ @ num += 5
$ echo $num
```

13

```
$ @ num++
$ echo $num
```

14

```
$ @ num2 = 5
$ @ num = $num2 + 5
$ echo $num
```

10

```
$ @ num = 1
echo $nums[$num]
```

one

Внимание!

В приведенных примерах первый символ `$` — это приглашение командной оболочки. При написании сценариев его указывать не нужно!

Общий синтаксис при работе с числовыми переменными выглядит так:

```
@ переменная оператор выражение
```

Здесь: `оператор` — это один из операторов присваивания языка C: `=`, `+=`, `- =`, `*=`, `/=` или `%=`. `выражение` — это арифметическое выражение, которое строится по тем же правилам, что и в языке C. Ничего удивительного — оболочка tcsh предназначена для тех, кто знаком с языком C, и должна была облегчить создание сценариев C-программистам.

Теперь рассмотрим пример работы с массивами чисел:

```
$ set a = (0 0 0 0 0)
$ @ a[1] = 10
$ @ a[3] = ($ages[1] + 5)
$ echo $a[3]
```

15

12.2.2. Чтение ввода пользователя

Для чтения ввода пользователя используется конструкция `"$<"`, например:

```
echo -n "Введите строку: "
set line = "$<"
```

12.2.3. Переменные оболочки, модификаторы форматов

При создании сценариев на tcsh вы можете использовать переменные оболочки, представленные в табл. 12.1. В табл. 12.2 представлены модификаторы формата командной строки, а в табл. 12.3 — модификаторы формата времени.

Таблица 12.1. *Переменные оболочки tcsh*

Переменная	Описание
`argv`	Массив содержит аргументы командной строки (параметры, переданные сценарию). `argv[1]` — первый параметр, а `argv[0]` — это имя самого сценария. Обратиться к параметрам можно через конструкцию `argv[n]` или `$n` — например, `argv[1]` или `$1`. Элемент массива `argv[*]` содержит все параметры вместе
`autolist`	Контролирует завершение команд и переменных. См. `man tcsh`
`autologout`	Включает возможность автоматического выхода. По умолчанию — 60 минут, если вы суперпользователь. Данная переменная не устанавливается (по умолчанию) для обычных пользователей

Таблица 12.1 *(окончание)*

Переменная	Описание
cdpath	Влияет на команду cd, задает путь поиска команд, обычно устанавливается в файле ~/.login, например: `set cdpath = (/home/denix /home/denix/bin)`
cwd	Содержит имя текущего каталога
fignore	Содержит массив суффиксов, которые tcsh будет игнорировать при завершении имен файлов
gid	Содержит идентификатор группы
histfile	Переменная содержит полное имя файла, в котором находится история команд. По умолчанию ~/.history
history	Размер списка истории
home или HOME	Имя домашнего каталога пользователя
mail	Содержит имя файлов и каталогов для проверки почты. TC Shell каждые 10 минут проверяет почту
owd	Предыдущий рабочий каталог
path или PATH	Путь поиска программ, обычно устанавливается в конфигурационных файлах tcsh: `set path = ( /usr/bin /bin /usr/local/bin /usr/bin/X11 ~/bin . )`
prompt	Содержит формат приглашения командной строки, аналогична переменной PS1 в bash. Модификаторы формата описаны в табл. 12.2. Значение по умолчанию: `set prompt = '! $ '`
prompt2	Содержит формат приглашения командной строки для управляющих структур while и foreach
prompt3	Содержит формат приглашения командной строки при проверке правописания
savehist	Количество команд, которые будут сохранены в файл истории
shell	Полный путь к исполнимому файлу оболочки
status	Код завершения последней выполненной команды
tcsh	Версия tcsh
time	Может содержать только число или же буквы и цифры. Если переменная содержит только число, то это количество секунд процессорного времени. Если выполнение какой-то команды заняло больше времени, чем указано в time, то после выполнения команды будет выведено, сколько времени она занимала. Если `time = 0`, то после каждой команды будет выведено время выполнения. Если time содержит буквы и цифры (табл. 12.3), то это просто формат времени
user	Имя пользователя

Таблица 12.2. *Формат командной строки*

Модификатор	Описание
%/	Текущий каталог
%~	То же, что и %/, но заменяет путь к домашнему каталогу пользователя тильдой
%! или %h или !	Текущий номер события
%m	Имя узла без домена
%M	Полное имя узла с доменом
%n	Имя пользователя
%t	Время дня
%p	Время дня (с секундами)
%d	День недели
%D	День месяца
%W	Месяц, формат мм
%y	Год, формат гг
%Y	Год, формат гггг
%#	Решетка (#) для суперпользователя или знак больше (>) для обычного пользователя
%?	Код завершения последней команды

Таблица 12.3. *Формат времени*

Модификатор	Описание
%U	Время, проведенное командой в пользовательском режиме (в процессорных секундах)
%S	Время, проведенное командой в режиме ядра (в процессорных секундах)
%W	Сколько раз процесс команды был выгружен на диск
%X	Средний размер сегмента кода программы, в килобайтах
%D	Средний объем памяти, используемый командой, в килобайтах
%K	Общий размер памяти, занятый командой (считается как %X+%D), в килобайтах
%M	Максимальный объем памяти, занятый командой, в килобайтах
%F	Количество ошибок страниц памяти
%I	Количество операций ввода
%O	Количество операций вывода

12.2.4. Управляющие структуры

Перед тем как приступить к рассмотрению управляющих структур, рассмотрим применение скобок в tcsh. Предположим, существует переменная:

```
$ set aa=abra
```

И нам нужно вывести ее в составе строки. Для этого мы будем использовать фигурные скобки:

```
$ echo ${aa}cadabra
```

abracadabra

Условный оператор *if*

Синтаксис оператора `if` очень прост:

```
if (выражение) команда
```

Команда будет выполнена, если выражение истинно. Выражения формируются так же, как в языке C. В листинге 12.6 представлен небольшой сценарий, проверяющий количество аргументов, переданных ему.

Листинг 12.6. Первый сценарий на tcsh

```
#!/bin/tcsh
if ( $#argv == 0 ) echo "Аргументы не заданы"
```

Также можно использовать следующее выражение:

```
-n имя_файла
```

В данном случае возможные варианты `n` представлены в табл. 12.4.

Таблица 12.4. *Значения n*

n	Описание
b	Файл является блочным устройством (обмен данными с устройством осуществляется блоками данных)
c	Файл является символьным устройством (обмен данными с устройством осуществляется посимвольно)
d	Файл является каталогом
e	Файл существует
g	Для файла установлен бит SGID *(см. разд. 14.5.3)*
k	Для файла установлен "липкий" бит
l	Файл является символической ссылкой
o	Файл принадлежит текущему пользователю
p	Файл является именованным потоком (FIFO)
r	У пользователя есть право чтения файла

Таблица 12.4 *(окончание)*

n	Описание
s	Файл не пустой (ненулевой размер)
S	Файл является сокетом
t	Дескриптор файла открыт и подключен к экрану
u	Для файла установлен SUID *(см. разд. 14.5.3)*
w	У пользователя есть право записи файла
x	У пользователя есть право выполнения файла
X	Файл является встроенной командой или его исполнимый файл найден при поиске в каталогах, указанных в `$path`
z	Файл пуст (нулевой размер)

А вот пример использования данного условия:

```
if -e $1 echo "Файл существует"
```

Условный оператор *if...then...else*

Условный оператор *if...then…else* похож на *if*, только к нему добавлен блок *else* (иначе), который выполняет команды в случае ложности условия.

Сокращенная версия оператора выглядит так:

```
if (выражение) then
    команды, которые будут выполнены в случае истинности выражения
endif
```

Полная версия оператора выглядит так:

```
if (выражение) then
    команды, которые будут выполнены в случае истинности выражения
else
    команды, которые будут выполнены в противном случае (когда выражение =
false)
endif
```

Существует еще одна форма этого оператора, точнее, это отдельный оператор *if...then...elif*:

```
if (выражение1) then
     команды (если выражение1 = true)
else if (выражение2) then
    команды (если выражение2 = true)
. . .
else
     команды (если ни одно из выражений не равно true)
endif
```

Рассмотрим небольшой пример использования условного оператора (листинг 12.7).

Листинг 12.7. Пример использования условного оператора

```
#!/bin/tcsh
# Получаем число из командной строки
set num = $argv[1]
set flag
#
if ($num < 0) then
    @ flag = 1
else if (0 <= $num && $num < 50) then
    @ flag = 2
else if (50 <= $num && $num < 1000) then
    @ flag = 3
else
    @ flag = 4
endif
#
echo "Flag: ${flag}."
```

Оператор *foreach*

Оператор *foreach* удобно использовать для перебора массивов, его синтаксис:

```
foreach индекс-цикла (список-аргументов)
     команды
end
```

Цикл `foreach` удобно использовать и при работе с файлами, например:

```
foreach f ( *.txt )
   echo $f
end
```

Здесь мы в цикле выводим имена всех текстовых файлов в текущем каталоге. Конечно, проще использовать для этой цели команду `ls`, но здесь мы просто продемонстрировали использование `foreach`.

Оператор *while*

Оператор `while` — это еще один вариант цикла. Команды, находящиеся в теле цикла, выполняются до тех пор, пока выражение истинно:

```
while (выражение)
    команды
end
```

Рассмотрим пример сценария, вычисляющего факториал числа *n*, при этом *n* задается в командной строке (листинг 12.8).

Листинг 12.8. Пример использования while

```
#!/bin/tcsh
set n = $argv[1]
set i = 1
set fact = 1
#
while ($i <= $n)
    @ fact *= $i
    @ i++
end
#
echo "Факториал числа $n равен $fact"
```

В циклах вы можете использовать также операторы `break` и `continue`. Оператор `break` прерывает цикл и передает управление оператору, следующему за `end`. Оператор `continue` прерывает только текущую итерацию цикла и передает управление оператору `end`, который после получения управления начнет следующую итерацию цикла.

Оператор *switch*

В ряде случаев использовать оператор `switch` намного удобнее, чем серию условных операторов. Синтаксис `switch` следующий:

```
switch (строка)
    case образец1:
        команды1
    breaksw
    case образец2:
        команды2
    breaksw
    ...
    default:
        команды по умолчанию
    breaksw
endsw
```

Вот пример использования оператора `switch`:

```
set string = test
switch (string)
       case test:
              echo "Строка: test"
       breaksw
       case text:
              echo "Строка: text"
       breaksw
       default:
              echo "Строка не опознана"
       breaksw
endsw
```



Глава 13

24 полезные команды

13.1. Команда *man*: справочная система

В любой UNIX-подобной операционной системе очень много команд. На полках книжных магазинов можно найти даже целые справочники команд, в которых описаны все особенности той или иной команды. По сути, большинство из таких справочников — это систематизированные и переведенные на русский язык страницы руководства стандартной справочной системы man.

Man (от англ. *manual*) — это справочная система FreeBSD (есть она и в других UNIX-системах). Справочная система содержит информацию о каждой программе, установленной в вашей системе. Откуда система знает обо всех программах? В *разд. 8.2* мы уже рассматривали этот вопрос. Разработчики программ условились вместе с программой поставлять специальный файл справочной системы (MAN-файл). И лишь в очень редких случаях недобросовестные разработчики не выполняют это условие. Для получения справки о какой-либо программе нужно ввести команду:

```
man имя_программы
```

Страницу руководства можно листать вперед и назад, подобно просмотру документа в программе `less`. Для выхода из справочной системы достаточно нажать клавишу <Q>.

В *главах 8*, *11* и *12* мы уже познакомились с некоторыми командами, а в *главе 14* рассмотрим команды для работы с файловой системой. Здесь же мы поговорим о командах, которые пригодятся любому пользователю/администратору BSD-системы.

13.2. Команда *uname*: информация о системе

Команда `uname` без параметров выводит название операционной системы — FreeBSD, но это мы и без нее знаем. Однако, указав параметры (табл. 13.1), мы можем "выжать" из `uname` много полезной информации (рис. 13.1).

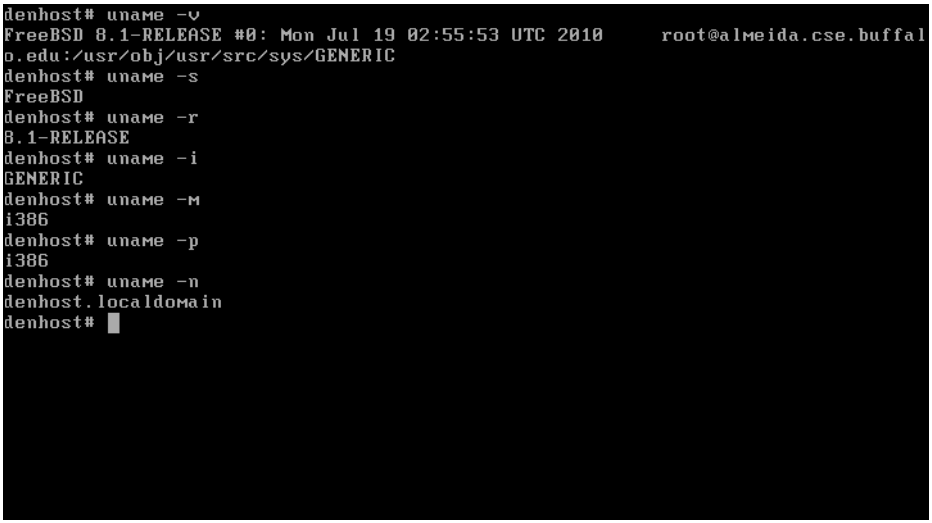


Рис. 13.1. Команда `uname`

Таблица 13.1. *Параметры команды uname*

Параметр	Описание
`-v`	Информация о релизе операционной системы. Выводится номер версии, ветка (RELEASE, STABLE, CURRENT), дата сборки
`-s`	Выводит название операционной системы на стандартный вывод (действие по умолчанию). Для FreeBSD будет выведено **FreeBSD**
`-r`	Информация только о релизе, то есть номере версии и ветке (RELEASE, STABLE, CURRENT)
`-p` и `-m`	В Linux существует команда `arch`, выводящая информацию об архитектуре процессора. Но в BSD такой команды нет, вместо нее нужно использовать параметры `-p` и `-m` команды `uname`. Первый выводит тип процессора, второй — тип аппаратной платформы. Если вы установили FreeBSD для платформы i386 на процессор этой же платформы, то вывод обоих параметров будет идентичен. Но ведь вы можете установить FreeBSD для i386 на машину с архитектурой amd64 — работать система будет, но не будут использоваться все преимущества платформы amd64. В этом случае вывод параметров будет различным
`-n`	Выводит имя узла и имя домена
`-i`	Информация о ядре
`-a`	Все, что предусмотрено параметрами `-m`, `-n`, `-r`, `-s` и `-v`

13.3. Команда *clear*: очистка экрана

Команда `clear` очищает экран при работе в консоли (терминале).

Пример использования:

```
$ clear
```

13.4. Команда *date*: вывод и установка даты и времени

Команда `date` служит для вывода текущей даты. Может применяться также для установки даты, если запущена от имени администратора.

Пример использования:

```
$ date
# date 0912171710
```

Первая команда выводит дату, а вторая команда — устанавливает дату (при условии, что команда запущена от имени root) 09 декабря (0912) 2010 года (10) и время 17:17. Как видите, установка даты осуществляется в формате MMddhhmmYY (MM — месяц, dd — число, hh — часы, mm — минуты, YY — год).

Команда `date` может вывести дату в указанном вам формате. Для знакомства со всеми форматами даты введите команду `man date`.

13.5. Команда *exit*: выход из оболочки

С этой командой `exit` мы уже знакомы. Она используется для выхода из оболочки. Если используемая в данный момент оболочка является оболочкой по умолчанию, происходит выход из системы (завершение сеанса).

13.6. Команда *passwd*: изменение пароля

Команда обеспечивает изменение пароля пользователя, который ее запустил. Суперпользователь root имеет право изменить пароль любого пользователя так:

```
# passwd имя_пользователя
```

13.7. Команда *uptime*: информация о работе системы

Команда `uptime` (рис. 13.2) выводит статистическую информацию о работе системы: сколько времени прошло с момента последней перезагрузки (собственно, это и есть время "uptime"), сколько пользователей в данный момент подключено к системе и среднюю загрузку системы за последние 1, 5 и 15 минут.

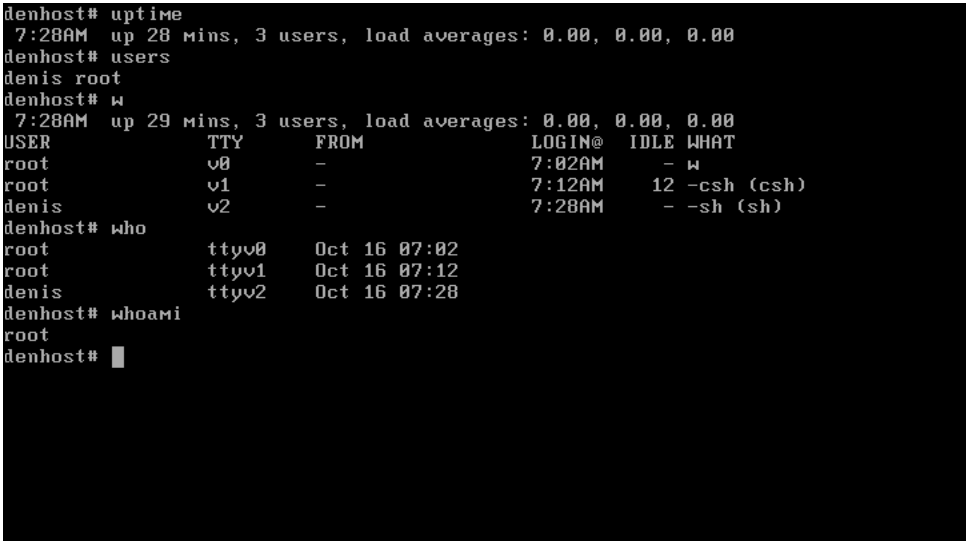


Рис. 13.2. Команды `uptime`, `users`, `w`, `who` и `whoami`

13.8. Команда *users*: информация о пользователях

Данная команда выводит пользователей, подключенных к системе в данный момент (см. рис. 13.2). Можно видеть, что в системе зарегистрировано два пользователя: denis и root. Но ведь способы входа в систему могут быть разными. Один пользователь может войти в систему удаленно — по SSH или FTP, другой может работать за консолью системы локально. Об этом команда `users` рассказать не может, зато могут команды `w` и `who`.

13.9. Команды *w*, *who* и *whoami*: подробная информация о пользователях

Эти три родственные команды (см. рис. 13.2) выводят следующую информацию:

- команда `w` — список пользователей, подключенных к системе; виртуальный терминал, с которого работает пользователь; хост, откуда пользователь вошел в системы (**FROM**) — только для удаленных сеансов; время входа в систему для каждого пользователя, статистику использования системы (**IDLE** — время простоя), выполняемые каждым пользователем задачи;
- команда `who` — список пользователей, подключенных к системе; виртуальный терминал; время и дату входа каждого пользователя;
- команда `whoami` — имя пользователя, который ввел команду.

13.10. Команда *diff*: сравнение файлов

Команда используется для сравнения двух файлов. Формат вызова программы `diff` такой:

```
diff параметры файл1 файл2
```

Результат сравнения выводится так: отличающиеся строки помечаются символами > и <. Строка из первого файла помечается символом <, а строка из второго файла — символом >.

Самые полезные параметры программы `diff` приведены в табл. 13.2.

Таблица 13.2. *Некоторые параметры программы* `diff`

Параметр	Описание
`-b`	Программа будет игнорировать пробельные символы в конце строки
`-B`	Игнорирует пустые строки
`-e`	Используется для создания сценария для редактора ed, который будет использоваться для превращения первого файла во второй
`-w`	Игнорирует пробельные символы
`-y`	Вывод в два столбца
`-r`	Используется для сравнения файлов в подкаталогах. Вместо первого файла указывается первый каталог, вместо второго файла указывается, соответственно, второй каталог

13.11. Команда *grep*: текстовый фильтр

Предположим, что у нас есть файл протокола /var/log/messages, и вы хотите вывести все сообщения, связанные с демоном pppd. Понятно, что вручную выделить все нужные сообщения будет довольно трудно. Но с помощью `grep` можно автоматизировать эту задачу:

```
cat /var/log/messages | grep ppp
```

Команда `cat /var/log/messages` передаст содержимое файла /var/log/messages на стандартный ввод программы `grep`, которая, в свою очередь, выделит строки, содержащие строку ppp.

13.12. Команды *more* и *less*: постраничный вывод

Большой текстовый файл намного удобнее просматривать с помощью программ `less` или `more`. Программа `less` удобнее, чем `more` (только потому, что позволяет просматривать текст в обратном направлении, что не может команда `more`), если она есть в вашей системе:

```
cat /var/log/messages | grep ppp | less
```

13.13. Команды *head* и *tail*: вывод начала и "хвоста" файла

Команда `head` выводит первые десять строк файла, а `tail` — последние десять. Вообще количество строк может регулироваться с помощью параметра `-n`.

Примеры использования:

```
head -n 10 /var/log/messages
tail -n 15 /var/log/messages
```

13.14. Команда *wc*: подсчет слов, строк и символов в файле

Команда `wc` используется для подсчета слов в текстовом файле, для подсчета количества строк (если задан параметр `-l`) и символов (с параметром `-c`).

Пример использования:

```
wc /var/log/messages
wc -l /var/log/messages
wc -c /var/log/messages
```

13.15. Команда *ftp*: стандартный FTP-клиент

Для открытия соединения с любым FTP-сервером введите команду:

```
ftp <имя или адрес FTP-сервера>
```

Можно просто ввести команду `ftp`, а в ответ на приглашение

ftp>

ввести команду:

```
open <имя или адрес FTP-сервера>
```

Лично мне больше нравится первый вариант, поскольку он позволяет сэкономить время. При подключении к серверу вы сможете ввести имя пользователя и пароль:

[den@dhsilabs ~]$ `ftp`
ftp> `open ftp.narod.ru`
Connected to ftp.narod.ru.
220 first-ftp.narod.ru (Libra FTP daemon 0.17 20100219)
Name (ftp.narod.ru:den): `den`
331 Password required
Password:
230 Logged in, proceed
Remote system type is UNIX.
ftp>

Подключившись к серверу, вы можете ввести команду `help`, чтобы просмотреть список доступных команд. Для получения справки по той или иной команде введите `help <имя_команды>` (рис. 13.3). Наиболее популярные команды FTP-клиента приведены в табл. 13.3.

```
ftp> help
Commands may be abbreviated.  Commands are:

!               features        mls             prompt          site
$               fget            mlsd            proxy           size
account         form            mlst            put             sndbuf
append          ftp             mode            pwd             status
ascii           gate            modtime         quit            struct
bell            get             more            quote           sunique
binary          glob            mput            rate            system
bye             hash            mreget          rcvbuf          tenex
case            help            msend           recv            throttle
cd              idle            newer           reget           trace
cdup            image           nlist           remopts         type
chmod           lcd             nmap            rename          umask
close           less            ntrans          reset           unset
cr              lpage           open            restart         usage
debug           lpwd            page            rhelp           user
delete          ls              passive         rmdir           verbose
dir             macdef          pdir            rstatus         xferbuf
disconnect      mdelete         pls             runique         ?
edit            mdir            pmlsd           send
epsv4           mget            preserve        sendport
exit            mkdir           progress        set
ftp>
```

Рис. 13.3. Список команд FTP-клиента

Таблица 13.3. *Некоторые команды FTP-клиента*

Команда	Описание
`ls`	Вывод содержимого каталога
`get`	Загрузить файл с сервера
`put`	Загрузить файл на сервер
`mget`	Получить несколько файлов с сервера. Допускается использование масок файлов, например, `*.tgz`
`mput`	Загрузить несколько файлов на сервер
`cd`	Изменить каталог
`mkdir`	Создать каталог
`rmdir`	Удалить пустой каталог
`delete`	Удалить файл

13.16. Команды *links* и *lynx*: текстовые браузеры

Если графический режим недоступен (например, на сервере), а по сети побродить хочется, можно использовать текстовые браузеры `links` и `lynx`. По умолчанию эти браузеры не устанавливаются, но их можно легко установить из портов:

```
# cd /usr/ports/www/links
# make install clean
# cd /usr/ports/www/lynx
# make install clean
```

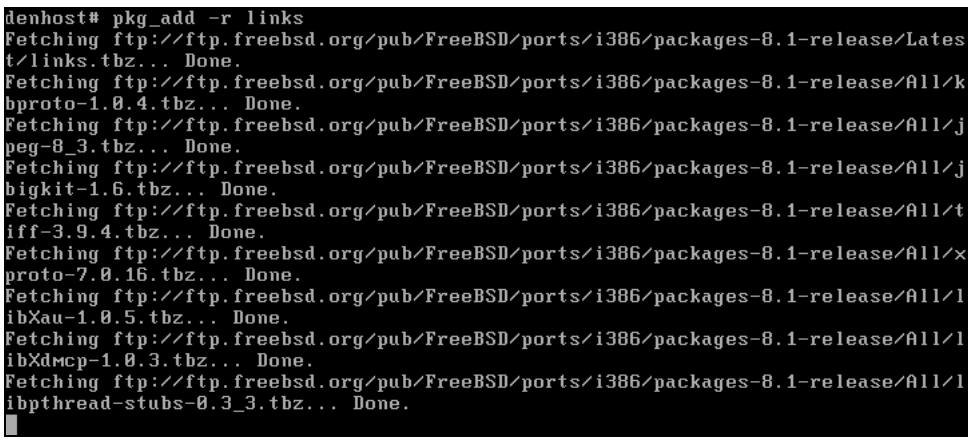


Рис. 13.4. Процесс установки браузера `links`



Рис. 13.5. Браузер `links`

Можно использовать также команду `pkg_add` (рис. 13.4) — более быстрый способ установки, например:

```
# pkg_add -r links
```

На рис. 13.5 изображен браузер `links`, в который загружен мой сайт **http://www.dkws.org.ua**. На момент создания скриншота моя система еще не была русифицирована, поэтому не обращайте внимания на транслитерацию. Кстати, это очень полезная особенность `links` — даже если вы не успели русифицировать консоль, вы все равно сможете просматривать русскоязычные сайты.

13.17. Команда *md5*: вычисление контрольного кода MD5

С целью проверки подлинности некоторых файлов, передаваемых через Интернет, используется алгоритм MD5, точнее, контрольный код, вычисленный с использованием этого алгоритма. Разработчик программы выкладывает в Интернете пакет со своей программой и на своем сайте публикует контрольный код. Вы скачиваете пакет и вычисляете его контрольный код. Если коды отличаются, то файл при передаче был поврежден (или это другая версия пакета, которая, возможно, была подложена злоумышленником с целью установки вражеского кода в вашу систему).

Использовать программу нужно так: `md5 файл`

13.18. Команда *df*: информация об использовании дискового пространства

Команду `df` удобно использовать с параметром `-h`, который обеспечивает вывод в удобном для человека формате, то есть в мегабайтах, гигабайтах, а не блоках. Сравните вывод команд `df` и `df -h` (рис. 13.6). Правда, с параметром `-h` удобнее? Вывод `df` ясен: показана файловая система, ее размер, использованное (**Used**) и доступное (**Avail**) пространство, процент использования файловой системы и точка монтирования.

```
denhost# df
Filesystem  1K-blocks    Used   Avail Capacity  Mounted on
/dev/ad0s1a    507630  172980  294040    37%    /
devfs               1       1       0   100%    /dev
/dev/ad0s1e    507630      16  467004     0%    /tmp
/dev/ad0s1f   9442528 3860142 4826984    44%    /usr
/dev/ad0s1d   1251790    4050 1147598     0%    /var
denhost# df -h
Filesystem     Size    Used   Avail Capacity  Mounted on
/dev/ad0s1a    496M    169M    287M    37%    /
devfs          1.0K    1.0K      0B   100%    /dev
/dev/ad0s1e    496M     16K    456M     0%    /tmp
/dev/ad0s1f    9.0G    3.7G    4.6G    44%    /usr
/dev/ad0s1d    1.2G    4.0M    1.1G     0%    /var
denhost#
```

Рис. 13.6. Команда `df`

13.19. Команда *free* своими руками

В Linux есть одна очень полезная команда — `free`, выводящая информацию об использовании оперативной памяти. Но в FreeBSD такая команда отсутствует. Сейчас мы попытаемся создать эту команду самостоятельно. В Интернете я нашел написанный на языке Perl сценарий, выводящий нужную нам информацию. Поскольку авторство принадлежит не мне, то вместо сценария я приведу только адреса, откуда его можно скачать:

- **http://itblog.su/wp-content/uploads/2009/01/freebsd-memorypl.txt**;
- **http://dkws.org.ua/files/freebsd-memorypl.txt**.

Скачаем и установим в нашей системе этот файл:

```
# fetch -o /usr/local/bin/free http://dkws.org.ua/files/freebsd-memorypl.txt
# chmod +x /usr/local/bin/free
# rehash
```

> ***Примечание***
>
> Команду `rehash` нужно вводить только пользователям оболочки tcsh, если у вас bash или просто sh, эту команду вводить не нужно.

Процесс установки файла freebsd-memorypl.txt показан на рис. 13.7, а на рис. 13.8 — вывод команды `free`. Обратите внимание, как FreeBSD экономно использует оперативную память — из 256 Мбайт занять всего 59 Мбайт.

Кроме указанного способа получить вариант команды `free` можно, установив порт `freecolor`:

```
# cd /usr/ports/sysutils/freecolor
# make install clean
```

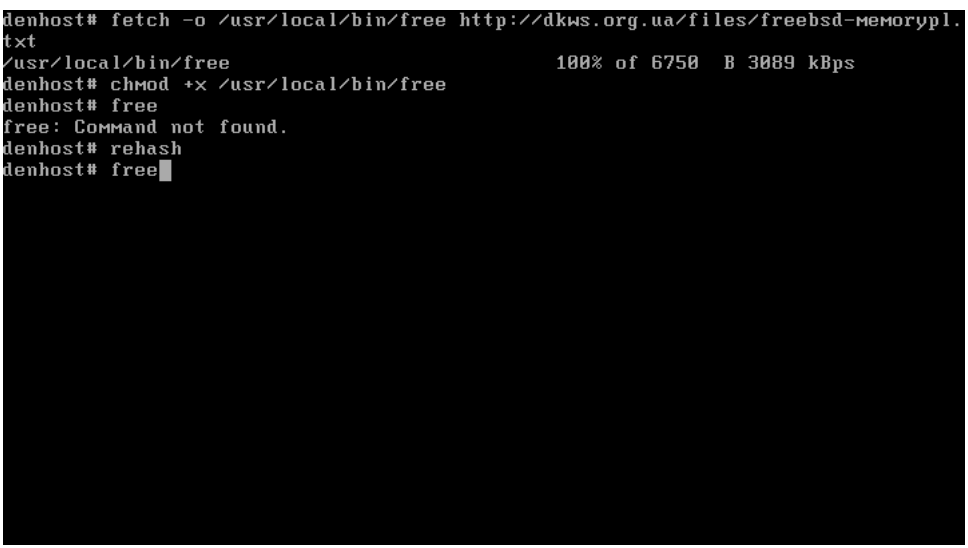


Рис. 13.7. Установка freebsd-memorypl.txt

```
SYSTEM MEMORY INFORMATION:
mem_wire:          35336192 (     33MB) [ 14%] Wired: disabled for paging out
mem_active:   +     8495104 (      8MB) [  3%] Active: recently referenced
mem_inactive:+     31879168 (     30MB) [ 12%] Inactive: recently not referenced

mem_cache:    +        20480 (      0MB) [  0%] Cached: almost avail. for allocat
ion
mem_free:     +    174116864 (    166MB) [ 69%] Free: fully available for allocat
ion
mem_gap_vm:   +       389120 (      0MB) [  0%] Memory gap: UNKNOWN
-------------- ------------ ----------- ------
mem_all:      =    250236928 (    238MB) [100%] Total real memory managed
mem_gap_sys: +      4993024 (      4MB)        Memory gap: Kernel?!
-------------- ------------ -----------
mem_phys:     =    255229952 (    243MB)        Total real memory available
mem_gap_hw:   +     13205504 (     12MB)        Memory gap: Segment Mappings?!
-------------- ------------ -----------
mem_hw:       =    268435456 (    256MB)        Total real memory installed

SYSTEM MEMORY SUMMARY:
mem_used:          62418944 (     59MB) [ 23%] Logically used memory
mem_avail:    +    206016512 (    196MB) [ 76%] Logically available memory
-------------- ------------ ----------- ------
mem_total:    =    268435456 (    256MB) [100%] Logically total memory
denhost#
```

Рис. 13.8. Команда `free` в FreeBSD

```
denhost# freecolor
Physical  : [################################.......] 80%   (201500/249248)
Swap      : [####################################] 100%  (490304/490304)
denhost# freecolor -t -m -o
             total       used       free     shared    buffers     cached
Mem:           243         46        196          0          0          0
Swap:          478          0        478
Total:         722 = (      46 (used) +      675 (free))
denhost#
```

Рис. 13.9. Команда `freecolor` без параметров и с параметрами `-t`, `-m`, `-o`

Можно также установить сразу пакет freecolor:

```
# pkg_add -r freecolor
```

`freecolor` — это "разноцветная" версия команды `free`. Конечно, она мало чем похожа на команду `free` в Linux, но если мы укажем параметры `-t`, `-m` и `-o`, вывод ее будет напоминать вывод команды `free` в Linux (рис. 13.9).



ЧАСТЬ IV

Администрирование системы



Глава 14

Файловая система

14.1. Файловые системы, поддерживаемые FreeBSD

14.1.1. Производительность файловых систем

Все файловые системы для любой UNIX-совместимой операционной системы можно разделить на две группы: родные (native) и прочие. К родным относятся файловые системы, которые могут выступать в роли корневой файловой системы. Ко второй группе — все остальные файловые системы. Поддержка прочих файловых систем неизбежна — как минимум нужно поддерживать файловую систему CD9660 (для чтения с компакт-дисков) и VFAT/FAT32 (для чтения/записи флешки). Конечно, флешку можно переформатировать в родную файловую систему, но тогда универсальность этого носителя информации потеряется.

Несколько неоднозначна эта ситуация в Linux. Родными для нее являются файловые системы ext2, ext3 и ext4, но та же ReiserFs также подходит для монтирования в качестве корневой файловой системы Linux. С одной стороны, ReiserFS никогда не была родной для нее файловой системой, поскольку ее поддержка в Linux появилась существенно позже, чем сама Linux, с другой стороны — корневая файловая система Linux может находиться на разделе, отформатированном как ReiserFS.

С FreeBSD (OpenBSD, NetBSD) все чуть проще: родными для нее являются файловые системы UFS (UNIX File System), UFS2 (64-разрядная версия UFS) и, с недавнего времени, ZFS (Zettabyte File System), поддержка которой впервые появилась в FreeBSD 7.0, а, начиная с восьмой версии, уже больше не является экспериментальной.

Начнем с обычной UFS. Для этого придется вернуться лет на десять назад. Тогда в Linux господствовала файловая система ext2: обычная ext безнадежно устарела, ext3 была еще на стадии разработки и утверждения, XFS была портирована на Linux только в 2001 году, ReiserFS только-только была включена в ядро, а JFS еще не была настолько распространена. Другими словами, у Linux-пользователя альтернативы не было и приходилось использовать только ext2.

В FreeBSD тогда господствовала такая же ситуация, но с файловой системой UFS. Альтернатив не было. А производительность UFS в то время (2000–2002 годы) на фоне ext2 оставляла желать лучшего. При всех своих преимуществах FreeBSD

при операциях с файлами выглядела неповоротливым монстром. С появлением механизма SoftUpdates (далее SU) для UFS ситуация стала лучше: пара UFS+SU почти догоняла ext2 и обеспечивала примерно такую же производительность, как ReiserFS того времени (которая была тогда еще достаточно сырой).

> ***SOFTUPDATES***
>
> SoftUpdates — механизм, обеспечивающий восстановление целостности данных после краха системы. Альтернатива журналируемым файловым системам в Linux. Подробнее о SoftUpdates можно прочитать по адресу: **http://en.wikipedia.org/wiki/Soft_updates**.

В пятой версии FreBSD появилась файловая система UFS2. В нее было добавлено множество преимуществ. Самые основные заключались в 64-разрядности, благодаря чему поддерживались большие разделы, в фоновой проверке после сбоев и т. д. Но по производительности лидером оставалась ext2. Скажу больше — UFS (самая обычная UFS) и то работала немного быстрее, чем UFS2. Да и связка UFS2+SU проиграла всем файловым системам. В некоторых тестах хуже была только JFS.

Несколько улучшить производительность UFS2 можно двумя способами: монтировать ее в асинхронном режиме (добавить опцию `async` в файл /etc/fstab) или организовать программный RAID-массив с помощью драйвера CCD *(см. главу 20)*. Но эти способы имеют и свои недостатки. В первом случае снижается надежность (работа в асинхронном режиме чревата потерей данных), а во втором — понадобится как минимум два жестких диска. К тому же при монтировании в асинхронном режиме нельзя включить SU. Но даже если мы пойдем на потери (либо надежности, либо финансов — при покупке второго жесткого диска), производительность UFS2 все равно будет ниже, чем у ext2 без всяких "извращений".

Тем не менее, в 2005 году файловой системе UFS2 повезло. Получили широкое распространение SATA-винчестеры. Оказалось, что на таком оборудовании UFS2 способна раскрыть свой потенциал. Если сравнивать производительность UFS на обычном винчестере EIDE (ATA) и на SATA-винчестере, то в последнем случае прирост производительности составил 100% — то есть производительность возросла в два раза! Конечно, производительность ext2 и ext3 тоже повысилась, но не так сильно.

Вскоре произошло еще одно событие — в FreeBSD появилась поддержка ZFS, которая является не только файловой системой, но и менеджером томов. О самой ZFS мы поговорим отдельно, а пока остановимся на производительности.

При копировании множества мелких файлов лидером является ZFS — она более чем в два раза быстрее, чем UFS2. Самой медленной в этом случае является ext3, а файловые системы ext2 и ReiserFS находятся примерно на одинаковом уровне (но обе быстрее, чем ext3). Итак, первое место — ZFS, второе — UFS2, третье — ext2 и ReiserFS, четвертое — ext3.

При копировании файлов среднего размера (от 1 до 10 Мбайт) быстрее всех — ext3, второе место — файловые системы UFS2 и ReiserFS (примерно на одинаковом уровне), третье место — ext2, последнее место — ZFS. Тут полный провал, поскольку ZFS в этом тесте оказалась в два раза медленнее, чем ext3.

Теперь проведем третий тест — копирование больших файлов, таких как ISO-образы или видеофайлы (размер файла от 600 Мбайт). Самая быстрая — ZFS, затем с небольшим отрывом следуют файловые системы ext3 и ReiserFS (второе место), третье место — UFS2, четвертое место — ext2.

Примечание

Получилось, что UFS2 быстрее, чем ext2. Но ранее было сказано, что ext2 намного быстрее, чем UFS. Все верно. Десять лет назад, на старом "железе", ext2 была в лидерах. А вот при тестировании на современном оборудовании (двухъядерный процессор Intel, винчестер SATA-II, 4 Гбайт оперативной памяти) UFS2 работает намного шустрее.

Таким образом, в двух тестах из трех ZFS заняла первое место. А эти тесты являются самыми важными: копирование множества мелких файлов и одного огромного. Файлы среднего размера копируются реже, чем мелкие файлы и файлы большого размера. Например, резервная копия всех конфигурационных файлов системы будет произведена на ZFS быстрее, равно как и копирование огромного ISO-образа с последней версией FreeBSD.

Однако не следует использовать ZFS на старом оборудовании. Она достаточно требовательна к оперативной памяти, поэтому даже не пытайтесь поставить ее на компьютер с менее, чем 1 Гбайт оперативной памяти, — результат будет прямо противоположным. Вместо прироста производительности вы получите медленную и прожорливую систему.

14.1.2. Какую файловую систему выбрать?

В качестве корневой файловой системы я бы посоветовал вам выбрать UFS2. На новом оборудовании она не так уж и плоха, а ZFS старому оборудованию вообще противопоказана. При наличии мощного и современного "железа" можно выбрать ZFS, но на вашем месте я бы поосторожничал — совсем недавно ZFS была экспериментальной, может, пока лучше подождать?

Что же касается долгого разговора о производительности файловых систем, то приростом производительности UFS2 и ZFS обязаны, по сути, не каким-то особым своим свойствам, а новому оборудованию, которое сравняло шансы всех файловых систем.

14.1.3. Интересные факты о ZFS

Файловая система ZFS (Zettabyte File System) изначально была создана компанией Sun Microsystems для операционной системы Solaris. По сравнению с другими файловыми системами она поддерживает большие объемы данных, является одновременно и файловой системой, и менеджером логических томов, а также предоставляет простое управление томами хранения данных.

Можно долго говорить о преимуществах и недостатках ZFS, но народ требует зрелищ, а читатель — фактов, и еще лучше, когда эти факты представлены в цифрах и таблицах. Самые интересные факты о ZFS сведены в табл. 14.1.

Таблица 14.1. *Самые интересные факты о ZFS*

Параметр	Значение
Количество файлов в любой отдельной файловой системе	2^{48}
Максимальный размер файловой системы	16 эксабайт (2^{64} байт)
Максимальный размер одного файла	16 эксабайт (2^{64} байт)
Максимальный размер одного пула хранения (zpool)	3×10^{23} петабайт
Количество атрибутов файла	2^{48}
Количество файлов в каталоге	2^{48}
Количество устройств в любом пуле (zpool)	2^{64}
Количество пулов в системе	2^{64}
Количество файловых систем в пуле	2^{64}

В отличие от обычных файловых систем, которые размещаются на одном устройстве (если вам нужно работать с более чем одним устройством, приходится использовать менеджер томов), ZFS строится поверх виртуальных пулов хранения данных, которые называются zpool. Каждый пул состоит из виртуальных устройств (virtual devices — vdevs), каждое из которых может являться:

- физическим устройством;
- зеркалом (RAID 1) одного или нескольких устройств;
- группой из двух или более устройств (RAID Z, похож на RAID5, используется в ZFS).

Существуют разные версии ZFS и разные версии zpool. Просмотреть номер версии пула можно командой:

```
# zpool upgrade -v
```

```
This system is currently running ZFS pool version 10.
The following versions are supported:
VER  DESCRIPTION
---  --------------------------------------------------------
 1   Initial ZFS version
 2   Ditto blocks (replicated metadata)
 3   Hot spares and double parity RAID-Z
 4   zpool history
 5   Compression using the gzip algorithm
 6   bootfs pool property
 7   Separate intent log devices
 8   Delegated administration
 9   refquota and refreservation properties
 10  Cache devices
...
```

Версия пула 10, показанная в выводе, — далеко не новая, на данный момент существует уже 21-я версия пула. В этой книге ZFS не будет рассматриваться под-

робно, а заинтересовавшихся прошу посетить следующие страницы, где вы получите исчерпывающую информацию о ZFS:

- **http://docs.sun.com/app/docs/doc/820-0836/zfsover-1?l=ru&a=view**;
- **http://docs.sun.com/app/docs/doc/820-0836/zfsover-2?l=ru&a=view**;
- **http://hub.opensolaris.org/bin/view/Project+zfs-crypto/WebHome**;
- **http://www.sun.com/2004-0914/feature/**;
- **http://www.opensolaris.org/os/community/zfs/**.

14.1.4. Монтирование UFS2 в асинхронном режиме

Если вы ни разу не сталкивались с UNIX-подобной операционной системой, вам может быть непонятен термин "монтирование". Тогда пропустите этот раздел, пока не будет рассмотрена операция монтирования, после чего вы сможете вернуться к сюда уже с бóльшим пониманием.

В асинхронном режиме все операции ввода/вывода с файловой системой выполняются асинхронно, что позволяет увеличить производительность файловой системы, но при некотором неблагоприятном стечении обстоятельств может привести к потере данных.

Для включения асинхронного режима можно использовать опцию `-o async` команды `mount`. Если вы хотите использовать асинхронный режим постоянно, тогда добавьте опцию `async` в /etc/fstab, например:

```
/dev/ad0s1a    /       ufs     rw,async       1       1
```

После редактирования файла /etc/fstab нужно перезагрузить компьютер:

```
# reboot
```

Еще раз отмечу, что включение асинхронного режима ведет к снижению надежности файловой системы. К тому же при включенном механизме SoftUpdates опция `async` просто игнорируется. Поэтому вам надлежит выбирать: или производительность (опция `async`, без SoftUpfates), или надежность (с SoftUpdates).

Пользователям, выбравшим производительность, придется повышать надежность аппаратно. Вам понадобится хороший источник бесперебойного питания, гарантирующий автономную работу компьютера хотя бы в течение 20 минут, — чтобы вы успели обнаружить "потерю света", добраться до компьютера и штатно его выключить. А еще лучше купить "умный" ИБП, подающий на компьютер сигнал об отсутствии питания, после чего система сама инициирует завершение работы.

Эксперимента ради я включил асинхронный режим на своем сервере. Получил выигрыш в пять секунд при загрузке системы и довольно сомнительный выигрыш в почти десять секунд при копировании файла размером 602 Мбайт. Ради пяти секунд не хочется рисковать данными, поэтому, наигравшись, вернул все назад, как было.

14.1.5. Включение SoftUpdates

Администраторам стратегически важных серверов не до производительности — главное, чтобы сервер работал надежно. В этом случае вам необходимо включить механизм SoftUpdates. Для этого следует перезагрузиться в однопользовательский

(single) режим, размонтировать раздел, для которого вы хотите включить SoftUpdates, и ввести команду:

```
# tunefs -n enable [раздел]
```

Например:

```
# tunefs -n enable /home
```

Ждем выполнения команды `tunefs` и перезагружаем компьютер.

Следует напомнить, что на файловой системе UFS включение SoftUpdates приводит к некоторому повышению производительности и к тому же увеличивает надежность, поэтому мы убиваем двух зайцев сразу. Однако включение SoftUpdates не даст такого прироста производительности, как перевод файловой системы в асинхронный режим.

14.2. Особенности файловой системы FreeBSD

14.2.1. Обо всем сразу: слайсы, разделы, блоки, иноды

Слайсы и разделы

Как вы уже знаете из *главы 2*, жесткий диск разбивается на слайсы — это то, что в других операционных системах называется разделами или логическими дисками (томами).

Слайсы нумеруются с 1. Предположим, что на ATA-винчестере (устройство /dev/ad0) создан один-единственный логический диск — слайс с номером 1. Имя этого слайса будет выглядеть так: /dev/ad0s1. Просмотреть список слайсов можно командой:

```
# fdisk -p [имя_диска]
```

Например:

```
# fdisk -p /dev/ad0
```

BSD-слайсы, в отличие от разделов других файловых систем (Linux, Windows), делятся на BSD-разделы, или просто разделы:

- a — обычно используется для корневой файловой системы;
- b — для подкачки (swap);
- c — зарезервирован и никогда не задействуется в обычных условиях. В ряде случаев используется для адресации всего слайса (позже мы будем использовать этот раздел для создания программного RAID-массива);
- d — для каталога /var;
- e — для каталога /tmp;
- f — для каталога /usr.

Примечание

Всего на BSD-слайсе может быть восемь разделов: a и b заняты, как уже было отмечено, для корневой файловой системы и свопа. Раздел c используется для адресации всего слайса. Остальные разделы (d–h) необязательны и могут использоваться по усмотрению администратора.

Просмотреть список разделов можно командой:

```
# bsdlabel [слайс]
```

Например:

```
# bsdlabel /dev/ad0s1
```

Результат выполнения этой команды на моем компьютере представлен на рис. 14.1.

```
denhost# bsdlabel /dev/ad0s1
# /dev/ad0s1:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:  1048576        0    4.2BSD        0     0     0
  b:   980608  1048576      swap
  c: 25165287        0    unused        0     0         # "raw" part, don't edit

  d:  2586624  2029184    4.2BSD        0     0     0
  e:  1048576  4615808    4.2BSD        0     0     0
  f: 19500903  5664384    4.2BSD        0     0     0
denhost#
```

Рис. 14.1. Команда `bsdlabel`

Вообще-то, назначение разделов может быть и другим. Теоретически в слайсе могут быть всего два раздела: a — для корневой файловой системы и b — для подкачки. Некоторые администраторы отводят также раздел d — для каталога /home или вообще монтируют каталог /home к другому жесткому диску. Тут все зависит от предпочтений администратора.

Блоки данных

Каждый BSD-раздел состоит из суперблока, блока группы цилиндров (описывает группу цилиндров), таблицы инодов (inodes) и области блоков данных.

Область блоков данных занимает большую часть пространства каждой группы цилиндров. Она состоит из блоков файловой системы. Каждый блок файловой системы (логический блок) подобен физическому блоку (на которые делится жесткий диск).

Логический блок — минимальный квант информации, доступный за одну операцию чтения/записи. То есть за одну операцию чтения и записи система может прочитать или записать не менее одного блока данных. Логический блок данных не может быть меньше физического блока (который равен 512 байтам), но зато может быть больше — тогда его размер кратен целому числу физических блоков: 1024 байта (2 физических блока), 2048 байтов (четыре физических блока) и т. д. Чем больше размер логического блока, тем выше производительность, поскольку на копирование одного логического блока размером 1024 байта уйдет меньше времени, чем

на копирование двух блоков по 512 байтов. С точки зрения производительности файловой системы, чем больше размер блока, тем лучше.

Но всегда есть оборотная сторона медали. Чем выше размер блока, тем нерациональнее используется дисковое пространство. Представим, что у нас есть множество мелких файлов, занимающих всего несколько десятков байтов (50, 100, 150 байтов). Один файл в UFS не может занимать менее одного блока. То есть если размер блока равен 1024 байта, то для хранения файла размером 50 байтов все равно будет потрачено 1024 физических байта. Поэтому с точки зрения экономии дискового пространства, чем меньше размер блока, тем лучше. Но меньше 512 байтов он все равно быть не может.

Примечание

Ранее я специально уточнил, что в UFS один файл не может занимать менее одного файла. В ReiserFS один логический блок может быть разделен между несколькими файлами, что позволяет максимально экономить дисковое пространство.

Информационные узлы (иноды)

Теперь переходим к инодам (информационным узлам, i-узлам, или inodes в англ. литературе). Мы установили, что наш раздел поделен на логические блоки размером минимум 512 байтов каждый. В каждом блоке данных содержится какая-то информация. Откуда система знает, какой блок относится к тому или иному файлу? — ведь запись на жесткий диск идет не последовательно (как на магнитную ленту), а произвольно. Например, блоки с номерами 1, 2, 5 и 7 могут принадлежать файлу А, а блоки 3, 4, 6 — файлу Б.

Вот для решения этой задачи и используется индексная таблица, она же таблица индексных дескрипторов, или таблица инодов. Каждая запись в таблице inodes содержит следующую информацию:

- уникальный номер inode — каждому файлу присваивается свой уникальный номер inode, по сути inode — это счетчик файлов, создаваемых в файловой системе. При создании нового файла inode увеличивается на единицу;
- тип файла — обычный файл, файл устройства, именованный канал (pipe), сокет, символическая ссылка или каталог;
- размер файла — задается в таблице в байтах;
- количество ссылок на файл — если на файл есть хотя бы одна ссылка, он не может быть удален;
- адреса логических блоков — таблица инодов содержит адреса всех блоков, относящихся к файлу. Для нашего файла А здесь будут номера 1, 2, 5 и 7;
- число блоков — общее число блоков, занимаемых файлом;
- атрибуты файла — содержат информацию о владельце файла, правах доступа, времени создания/модификации и т. д.

Введите команду `ls -i /`, и вы увидите список файлов и каталогов корневой файловой системы и номера их инодов (рис. 14.2).

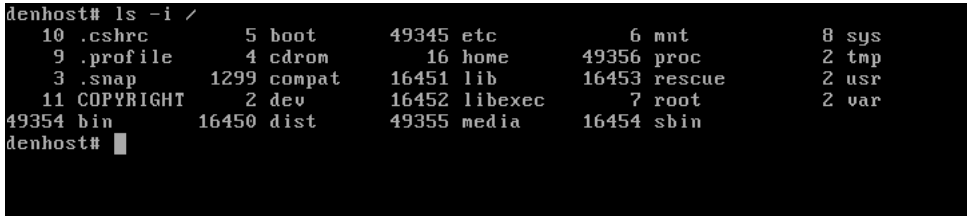


Рис. 14.2. Результат выполнения команды `ls -i /`

Если внимательно присмотреться, то можно заметить одну как бы неувязку. Каталогам tmp, usr, var и dev присвоен один и тот же номер инода — 2. А где же та самая уникальность? Давайте разберемся. Если бы у нас был один большой BSD-раздел, содержащий каталоги tmp, usr и var (то есть чтобы каталоги tmp, usr и var находились физически на одном и том же разделе), тогда этим каталогам были бы присвоены разные номера i-узлов. А если каталоги представляют собой точки монтирования (как в нашем случае), то им присваивается номер 2. Точка монтирования — это каталог, через который осуществляется доступ к данным, которые физически находятся на другом BSD-разделе, слайсе или даже на другом физическом жестком диске. А как же каталог dev? Каталог dev — это тоже, по сути, точка монтирования псевдофайловой системы устройств, которая находится в оперативной памяти.

Теперь, когда мы познакомились с внутренним устройством файловой системы, можно приступить к делам обыденным — поговорить об именах файлов, файлах устройств и прочих "виртуальностях".

14.2.2. Имена файлов в FreeBSD

По сравнению с Windows в FreeBSD несколько другие правила построения имен файлов, всем нам придется с этим смириться. Начнем с того, что в FreeBSD (да и в любой другой UNIX-системе) нет такого понятия, как *расширение* имени файла. В Windows, например, для файла Document1.doc именем файла является фрагмент Document1, а doc — это расширение. В FreeBSD Document1.doc — это имя файла, никакого расширения нет.

Максимальная длина имени файла в FreeBSD — 254 символа. Имя может содержать любые символы (в том числе и кириллицу), кроме / \ ? < > * " |. Но кириллицу в именах файлов я бы не рекомендовал вообще. Впрочем, если вы уверены, что не будете эти файлы передавать Windows-пользователям (на сменном носителе, по электронной почте), — используйте себе на здоровье. А при обмене файлами по электронной почте (кодировка-то у всех разная, поэтому вместо русскоязычного имени пользователь может увидеть абракадабру) имя файла лучше писать латиницей.

Придется также привыкнуть и к тому, что система FreeBSD чувствительна к регистру в имени файла: FILE.txt и FiLe.Txt — это два разных файла.

Разделение элементов пути осуществляется символом / (прямой слэш), а не \ (обратный слэш), как в Windows.

14.2.3. Файлы и устройства

Пользователи Windows привыкли к тому, что файл — это именованная область данных на диске. Отчасти так оно и есть. Отчасти — потому, что приведенное определение файла было верно для DOS (Disk Operating System) и Windows.

В FreeBSD же понятие файла значительно шире. Сейчас Windows-пользователи будут очень удивлены: в UNIX существуют файлы устройств, позволяющие обращаться с устройством, как с обычным файлом. Файлы устройств находятся в каталоге /dev (от devices). Да, через файл устройства мы можем обратиться к устройству! Если вы работали в DOS, то, наверное, помните, что что-то подобное было и там — существовали зарезервированные имена файлов: PRN (принтер), CON (клавиатура при вводе, дисплей при выводе), LPT*n* (параллельный порт, *n* — номер порта), COM*n* (последовательный порт).

> ***Примечание***
>
> Кому-то может показаться, что разработчики UNIX украли идею специальных файлов у Microsoft, но это не так. Наоборот, Microsoft позаимствовала идею файлов устройств из операционной системы UNIX, которая была создана еще до создания DOS (UNIX появилась в начале 70-х годов, а DOS — в начале 80-х годов прошлого века). Однако сейчас не время говорить об истории развития операционных систем, поэтому лучше вернемся к файлам устройств.

Вот самые распространенные примеры файлов устройств:

- /dev/ad*N* — файл жесткого диска с интерфейсами PATA (IDE) и SATA:
  - /dev/ad0 — жесткий диск, подключенный как Primary Master;
  - /dev/ad1 — жесткий диск, подключенный как Primary Slave;
  - /dev/ad2 — жесткий диск, подключенный как Secondary Master;
  - /dev/ad3 — жесткий диск, подключенный как Secondary Slave;
  - /dev/ad4 — жесткий диск, подключенный к первому SATA-разъему;
  - /dev/ad5 — не используется;
  - /dev/ad6 — жесткий диск, подключенный ко второму SATA-разъему;
  - /dev/ad7 — не используется;
  - /dev/ad8 — жесткий диск, подключенный к третьему SATA-разъему;
  - /dev/ad9 — не используется;
- /dev/ad*x*N — файл устройства раздела (слайса) на ATA-винчестере *x*, *N* — здесь номер раздела;
- /dev/da*N* — файл жесткого диска с интерфейсом SCSI или флешка (USB-диск);
- /dev/da*x*N — файл устройства раздела (слайса) на SCSI-винчестере;
- /dev/cd0 — привод CD/DVD, подключенный к SCSI-контроллеру;
- /dev/acd0 — привод CD/DVD, подключенный к ATA-контроллеру;
- /dev/fd*N* — дисковод для гибких дисков;
- /dev/cuau*N* — файл последовательного порта, *N* — номер порта (cuau0 соответствует COM1, cuau1 — COM2 и т. д.);
- /dev/ulpt*N* — принтер, подключенный к USB-порту, *N* — номер принтера.

У начинающих пользователей чаще всего вызывает непонимание образование имен жестких дисков. Вернемся к временам, когда еще не было SATA-дисков, а использовались только диски PATA (тогда они назывались IDE или EIDE) и SCSI. SCSI-диски назывались /dev/da*N*, где *N* — номер диска, зависящий от порядка подключения диска к SCSI-контроллеру. С PATA-дисками все тоже не вызывало сомнений. В обычной системе имеется два PATA-контроллера (не больше и не меньше — всего два), к каждому контроллеру можно подключить не более двух дисков. То есть в системе может существовать до четырех PATA-дисков. Имена дисков жестко определены:

- /dev/ad0 — жесткий диск, подключенный как Primary Master;
- /dev/ad1 — жесткий диск, подключенный как Primary Slave;
- /dev/ad2 — жесткий диск, подключенный как Secondary Master;
- /dev/ad3 — жесткий диск, подключенный как Secondary Slave.

Но вот появились диски SATA. И тут же возник вопрос: к какому классу дисков их отнести? В Linux SATA-диски были зачислены в ранг SCSI и получили имена /dev/sd*X*, как настоящие SCSI-диски. Но, как ни крути, SATA — это не SCSI, поэтому в FreeBSD их отнесли к PATA-дискам, имена тоже позаимствовали у PATA. Поскольку первые четыре имени зарезервированы для двух контроллеров PATA, то жесткий диск, подключенный к первому SATA-контроллеру, стал называться ad4. Но к SATA-контроллеру можно подключить только один жесткий диск. И раз уж диски SATA стали именоваться по аналогии с PATA, то имя /dev/ad5 должно быть зарезервировано под Slave-диск контроллера. Однако такого диска в природе не существует, поэтому имя ad5 никогда не используется. Аналогично с именами ad6–ad9.

Вернемся к файлам устройств. Они бывают двух типов: блочные и символьные. Обмен информации с блочными устройствами, например с жесткими дисками, осуществляется блоками информации, а с символьными — отдельными символами. Пример символьного устройства — последовательный порт.

14.2.4. Корневая файловая система и монтирование

Наверняка на вашем компьютере установлена система Windows. Выполните команду **Пуск | Компьютер** (рис. 14.3).

Скорее всего, вы увидите пиктограммы разделов жесткого диска (имена устройств C: и D:), пиктограмму привода CD/DVD (F:). Могли бы увидеть и пиктограмму гибкого диска (имя устройства — A:), но мой ноутбук не оснащен дисководом для гибких дисков, поэтому на рис. 14.3 диск A: отсутствует. Таким способом, с помощью буквенных обозначений A:, C:, D: и т. д., в Windows обозначаются корневые каталоги разделов жесткого диска и сменных носителей.

В FreeBSD (и в других UNIX-системах) существует понятие *корневой файловой системы*. Допустим, вы установили FreeBSD на слайс /dev/ad0s1. В разделе a этого слайса будет развернута корневая файловая система вашей BSD-системы. Корневой каталог обозначается прямым слэшем (/), то есть для перехода в корневой каталог в терминале (или консоли) нужно ввести команду `cd /`.

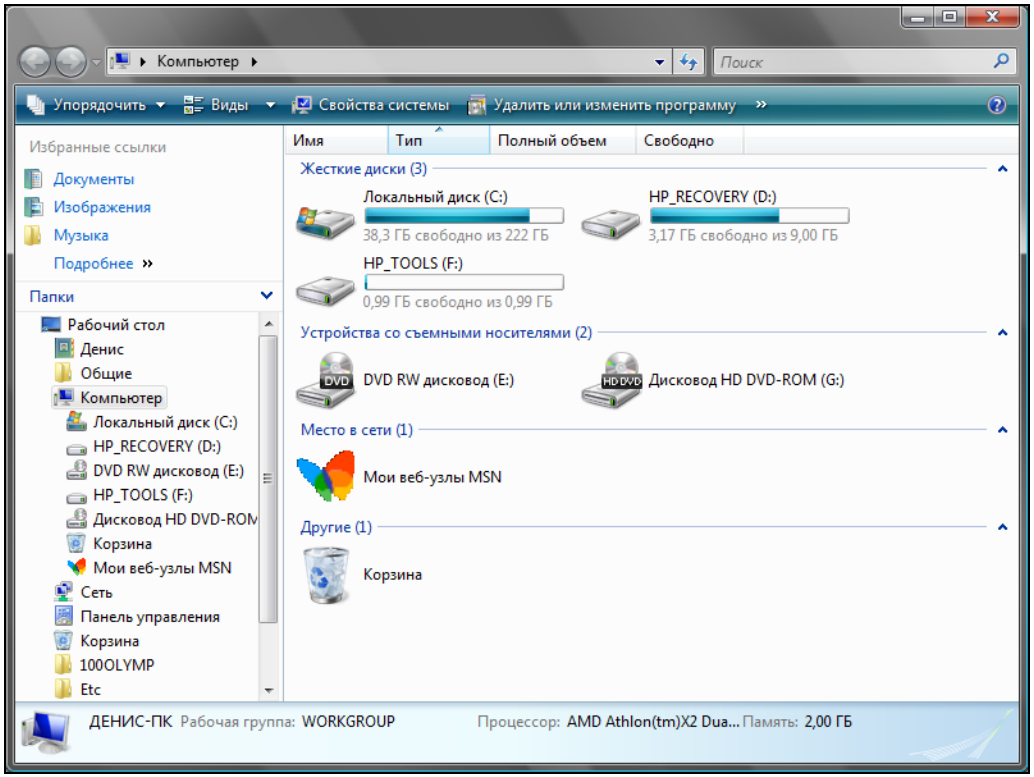


Рис. 14.3. Окно **Компьютер**

Понятно, что на вашем жестком диске есть еще слайсы. Чтобы получить доступ к этим разделам, вам нужно *подмонтировать* их к корневой файловой системе. После монтирования вы можете обратиться к содержимому разделов через точку монтирования — назначенный вами при монтировании специальный каталог, например /mnt. Монтированию файловых систем посвящен *разд. 14.6*, поэтому сейчас не будем говорить об этом процессе подробно.

14.2.5. Стандартные каталоги FreeBSD

Файловая система FreeBSD содержит следующие каталоги:

- / — корневой каталог;
- /bin — содержит стандартные программы (`cat`, `cp`, `ls` и т. д.);
- /boot — каталог загрузчика;
- /dev — содержит файлы устройств;
- /etc — содержит конфигурационные файлы системы;
- /home — содержит домашние каталоги пользователей (обычно это ссылка на /usr/home);
- /lib — библиотеки и модули;
- /media — содержит точки монтирования сменных устройств;

- /mnt — содержит точки монтирования временно смонтированных файловых систем (постоянно монтируемые файловые системы обычно монтируются к каталогам первого уровня: /usr, /var и т. д.);
- /opt — дополнительные пакеты программного обеспечения;
- /proc — каталог псевдофайловой системы procfs, предоставляющей информацию о процессах;
- /rescue — набор утилит, которые можно использовать для восстановления системы;
- /root — каталог суперпользователя root;
- /sbin — каталог системных утилит, выполнять которые имеет право пользователь root;
- /tmp — каталог для временных файлов;
- /usr — содержит пользовательские программы, документацию, исходные коды программ и ядра;
- /var — содержит постоянно изменяющиеся данные системы, например очереди системы печати, почтовые ящики, протоколы, замки и т. д.

Это основные каталоги, определенные стандартом FHS (Filesystem Hierarchy Standard)[1].

14.3. Команды для работы с файлами и каталогами

14.3.1. Работа с файлами

Здесь мы рассмотрим основные команды для работы с файлами в FreeBSD (табл. 14.2), а в последующих разделах этой главы — команды для работы с каталогами, ссылками и поговорим о правах доступа к файлам и каталогам.

Таблица 14.2. *Основные команды, предназначенные для работы с файлами*

Команда	Назначение
`touch <файл>`	Создает пустой файл
`cat <файл>`	Просмотр текстового файла
`tac <файл>`	Вывод содержимого текстового файла в обратном порядке, то есть сначала выводится последняя строка, потом предпоследняя и т. д.
`cp <файл1> <файл2>`	Копирует файл `<файл1>` в файл `<файл2>`. Если `<файл2>` существует, программа попросит разрешение на его перезапись
`mv <файл1> <файл2>`	Перемещает файл `<файл1>` в файл `<файл2>`. Эту же команду можно использовать и для переименования файла

[1] Подробнее о стандарте FHS можно прочитать по адресу: **http://rus-linux.net/MyLDP/file-sys/fhs-2.2-rus/index.html**.

Таблица 14.2 *(окончание)*

Команда	Назначение
`rm <файл>`	Удаляет файл
`locate <файл>`	Производит быстрый поиск файла
`which <программа>`	Выводит каталог, в котором находится программа, если она вообще установлена. Поиск производится в каталогах, указанных в переменной окружения `PATH` (это путь поиска программ)
`less <файл>`	Используется для удобного просмотра файла с возможностью скроллинга (постраничной прокрутки)

Рассмотрим небольшую серию команд (протокол выполнения этих команд приведен на рис. 14.4):

```
touch file.txt
echo "some text" > file.txt
cat file.txt
cp file.txt file-copy.txt
cat file-copy.txt
rm file.txt
cat file.txt
mv file-copy.txt file.txt
cat file.txt
```

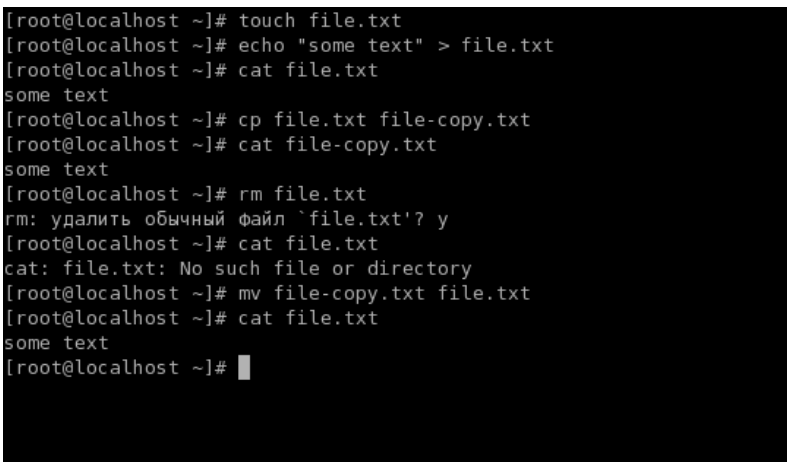


Рис. 14.4. Операции с файлом

Первая команда (`touch`) создает в текущем каталоге файл file.txt. Вторая команда (`echo`) записывает строку `some text` в этот же файл. Обратите внимание на символ > — это символ перенаправления ввода/вывода *(см. разд. 11.5)*.

Третья команда (cat) выводит содержимое файла — в файле записанная нами строка some text. Четвертая команда (cp) копирует файл file.txt в файл с именем file-copy.txt. После этого мы опять используем команду cat, чтобы вывести содержимое файла file-copy.txt, — надо же убедиться, что файл действительно скопировался.

Шестая команда (rm) удаляет файл file.txt. При удалении система спрашивает, хотите ли вы удалить файл. Если хотите удалить, то нужно нажать клавишу <Y>, а если нет, то <N>. Точно ли файл удален? Убедимся в этом: введите команду cat file.txt. Система нам сообщает, что нет такого файла.

Восьмая команда (mv) переименовывает файл file-copy.txt в файл file.txt. Последняя команда выводит исходный файл file.txt. Думаю, особых проблем с этими командами у вас не возникло, тем более что принцип действия этих команд вам должен быть знаком по командам DOS, которые, как квалифицированный пользователь Windows, вы должны знать наизусть.

Вместо имени файла иногда очень удобно указать *маску имени файла*. Например, у нас есть много временных файлов, имена которых заканчиваются фрагментом tmp. Для их удаления нужно воспользоваться командой: rm *tmp.

Если же требуется удалить все файлы в текущем каталоге, можно просто указать звездочку: rm *.

Аналогично, можно использовать символ ?, который, в отличие от звездочки, заменяющей последовательность символов произвольной длины, заменяет всего один символ. Например, нам нужно удалить все файлы, имена которых состоят из трех букв и начинаются на s:

```
rm s??
```

Будут удалены файлы s14, sqm, sr6 и т. д., но не будут тронуты файлы, имена которых состоят более чем из трех букв и которые не начинаются на s.

Маски имен можно также использовать и при работе с каталогами.

14.3.2. Работа с каталогами

Основные команды для работы с каталогами приведены в табл. 14.3.

Таблица 14.3. *Основные команды для работы с каталогами*

Команда	Описание
mkdir <каталог>	Создание каталога
cd <каталог>	Изменение каталога
ls <каталог>	Вывод содержимого каталога
rmdir <каталог>	Удаление пустого каталога
rm -r <каталог>	Рекурсивное удаление каталога

При указании имени каталога можно использовать следующие символы:

- . — означает текущий каталог. Если вы введете команду cat ./file, то она выведет файл file, который находится в текущем каталоге;

- `..` — родительский каталог. Например, команда `cd ..` переведет вас на один уровень вверх по дереву файловой системы;
- `~` — домашний каталог пользователя (об этом мы поговорим позже).

Теперь рассмотрим пример работы с каталогами на практике. Выполните следующие команды:

```
mkdir directory
cd directory
touch file1.txt
touch file2.txt
ls
cd ..
ls directory
rm directory
rmdir directory
rm -r directory
```

Первая команда (`mkdir`) создает каталог directory в текущем каталоге. Вторая команда (`cd`) переводит (изменяет каталог) в только что созданный каталог. Следующие две команды `touch` создают в новом каталоге два файла — file1.txt и file2.txt.

Команда `ls` без указания каталога выводит содержимое текущего каталога. Команда `cd ..` переводит в родительский каталог. Как уже было отмечено, в UNIX родительский каталог обозначается так: `..` (две точки), а текущий так: `.` (одна точка). То есть, находясь в каталоге directory, мы можем обращаться к файлам file1.txt и file2.txt без указания каталога или же так: `./file1.txt` и `./file2.txt`.

> ***Внимание!***
>
> Еще раз обратите внимание: в UNIX в отличие от Windows для разделения элементов пути используется прямой слэш (`/`), а не обратный (`\`)!

Кроме обозначений `..` и `.` в UNIX часто используется обозначение `~` (тильда) — это *домашний каталог*. Предположим, что наш домашний каталог /home/den. В нем мы создали подкаталог dir и поместили в него файл file1.txt. Полный путь к файлу можно записать так:

```
/home/den/dir/file1.txt
```

или же так:

```
~/dir/file1.txt
```

Как видите, тильда (`~`) заменяет часть пути. Удобно? Конечно!

Поскольку мы находимся в родительском для каталога directory каталоге, чтобы вывести содержимое только что созданного каталога в команде `ls`, нам нужно четко указать имя каталога:

```
ls directory
```

```
[root@localhost ~]# mkdir directory
[root@localhost ~]# cd directory
[root@localhost directory]# touch file.txt
[root@localhost directory]# touch file2.txt
[root@localhost directory]# ls
file2.txt  file.txt
[root@localhost directory]# cd ..
[root@localhost ~]# ls directory
file2.txt  file.txt
[root@localhost ~]# rm directory
rm: невозможно удалить каталог `directory': Is a directory
[root@localhost ~]# rmdir directory
rmdir: `directory': Directory not empty
[root@localhost ~]# rm -r directory
rm: спуститься в каталог `directory'? y
rm: удалить пустой обычный файл `directory/file.txt'? y
rm: удалить пустой обычный файл `directory/file2.txt'? y
rm: удалить Каталог `directory'? y
[root@localhost ~]#
```

Рис. 14.5. Операции с каталогами

Команда `rm` используется для удаления каталога. Но что мы видим — система отказывается удалять каталог! Пробуем удалить его командой `rmdir`, но и тут отказ. Система сообщает нам, что каталог не пустой, то есть содержит файлы. Для удаления каталога нужно удалить все файлы. Конечно, делать это не сильно хочется, поэтому проще указать опцию `-r` команды `rm` для рекурсивного удаления каталога. В этом случае сначала будут удалены все подкаталоги (и все файлы в этих подкаталогах), а затем будет удален сам каталог (рис. 14.5).

Команды `cp` и `mv` работают аналогично: для копирования (перемещения/переименования) сначала указывается каталог-источник, а потом каталог-назначение. Для каталогов желательно указывать параметр `-r`, чтобы копирование (перемещение) производилось рекурсивно.

14.4. Команда *ln*: создание ссылок

В UNIX допускается, чтобы один и тот же файл существовал в системе под разными именами. Для этого используются ссылки. Ссылки бывают двух типов: жесткие и символические. Жесткие ссылки жестко привязываются к файлу — вы не можете удалить файл, пока на него указывает хотя бы одна жесткая ссылка. А вот если на файл указывают символические ссылки, его удалению ничто не помешает.

Жесткие ссылки не могут указывать на файл, который находится за пределами файловой системы. Предположим, у вас два BSD-раздела: один корневой, а второй используется для домашних файлов пользователей и монтируется к каталогу /home корневой файловой системы. Так вот, вы не можете создать в корневой файловой системе ссылку, которая ссылается на файл в файловой системе, подмонтированной к каталогу /home. Это очень важная особенность жестких ссылок. Если вам нужно создать ссылку на файл, который находится за пределами файловой системы, вам следует использовать символические ссылки.

Для создания ссылок используется команда `ln`:

```
ln file.txt link1
ln -s file.txt link2
```

Первая команда создает жесткую ссылку link1, ссылающуюся на текстовый файл file.txt. Вторая команда создает символическую ссылку link2, которая ссылается на этот же текстовый файл file.txt.

Модифицируя ссылку (все равно какую — link1 или link2), вы автоматически модифицируете исходный файл — file.txt.

Особого внимания заслуживает операция удаления. По идее, если вы удаляете ссылку link2, файл file.txt также должен быть удален, но не тут-то было — вы не можете его удалить до тех пор, пока на него указывает хоть одна жесткая ссылка. При удалении ссылки link2 просто будет удалена символьная ссылка, но жесткая ссылка и сам файл останутся. Если же вы удалите ссылку link1, будет удален и файл file.txt, поскольку на него больше не ссылается ни одна жесткая ссылка.

14.5. Команды *chmod*, *chown* и *chattr*

14.5.1. Команда *chmod*: права доступа к файлам и каталогам

Для каждого каталога и файла вы можете задать права доступа. Точнее, права доступа автоматически задаются при создании каталога/файла, а вам при необходимости можно их изменить. Какая может быть необходимость? Например, вам нужно, чтобы к вашему файлу-отчету смогли получить доступ пользователи — члены вашей группы. Или вы создали обычный текстовый файл, содержащий инструкции командного интерпретатора. Чтобы этот файл стал сценарием, вам нужно установить право на выполнение для этого файла.

Существуют три права доступа: чтение (`r`), запись (`w`), выполнение (`x`). Для каталога право на выполнение означает право на просмотр содержимого каталога.

Вы можете установить разные права доступа для владельца (то есть для себя), для группы владельца (то есть для всех пользователей, входящих в одну с владельцем группу) и для прочих пользователей. Пользователь root может получить доступ к любому файлу или каталогу вне зависимости от прав, которые вы установили.

Чтобы просмотреть текущие права доступа, введите команду:

```
ls -l <имя файла/каталога>
```

Например,

```
ls -l video.txt
```

В ответ программа выведет следующую строку:

-r--r----- 1 den group 300 Apr 11 11:11 video.txt

В этой строке фрагмент -r--r----- описывает права доступа:

- первый символ — это признак каталога. Сейчас перед нами файл. Если бы перед нами был каталог, то первый символ был бы символом d (от directory);

- последующие три символа (r--) определяют *права доступа владельца файла или каталога*. Первый символ — это чтение, второй — запись, третий — выполнение. Как можно видеть, владельцу разрешено только чтение этого файла, запись и выполнение запрещены, поскольку в правах доступа режимы `w` и `x` не определены;
- следующие три символа (r--) задают *права доступа для членов группы владельца*. Права такие же, как и у владельца: можно читать файл, но нельзя изменять или запускать;
- последние три символа (---) задают *права доступа для прочих пользователей*. Прочие пользователи не имеют права ни читать, ни изменять, ни выполнять файл. При попытке получить доступ к файлу они увидят сообщение **Access denied**.

> ***Примечание***
>
> После прав доступа команда `ls` выводит количество жестких ссылок на файл, имя владельца файла, имя группы владельца, размер файла, дату и время создания, а также имя файла. Если файл является ссылкой, то после имени файла выводится название файла, на который ссылается ссылка: → `имя_файла или каталога`.

Права доступа задаются командой `chmod`. Существуют два способа указания прав доступа: *символьный* (когда указываются символы, задающие право доступа, — `r`, `w`, `x`) и *абсолютный*.

Так уж заведено, что в мире UNIX чаще пользуются абсолютным методом. Разберемся, в чем он заключается. Рассмотрим следующий набор прав доступа:

```
rw-r-----
```

Указанный набор прав доступа предоставляет владельцу право чтения и модификации файла (`rw-`), запускать файл владелец не может. Члены группы владельца могут только просматривать файл (`r--`), а все остальные пользователи не имеют вообще никакого доступа к файлу.

Возьмем отдельный набор прав, например для владельца: `rw-`

Чтение разрешено — мысленно записываем 1, запись разрешена — запоминаем еще 1, а вот выполнение запрещено, поэтому запоминаем 0. Получается число 110. Если из двоичной системы перевести число 110 в восьмеричную, получится число 6. Для перевода можно воспользоваться табл. 14.4.

Таблица 14.4. *Преобразование чисел из двоичной системы в восьмеричную*

Двоичная система	Восьмеричная система	Двоичная система	Восьмеричная система
000	0	100	4
001	1	101	5
010	2	110	6
011	3	111	7

Аналогично произведем разбор прав для членов группы владельца. Получится двоичное 100, то есть восьмеричное 9. С третьим набором (---) все вообще просто — это 000, то есть 0.

Записываем полученные числа в восьмеричной системе в порядке владелец-группа-остальные. Получится число 640 — это и есть права доступа. Для того чтобы установить эти права доступа, выполните команду:

```
chmod 640 <имя_файла>
```

Наиболее популярные права доступа:

- 644 — владельцу можно читать и изменять файл, остальным пользователям — только читать;
- 666 — читать и изменять файл можно всем пользователям;
- 777 — всем можно читать, изменять и выполнять файл.

> ***Примечание***
>
> Напомню, что для каталога право выполнения — это право просмотра оглавления каталога.

Иногда символьный метод оказывается проще. Например, у нас есть файл script, который нужно сделать исполнимым, для этого можно применить команду:

```
chmod +x script
```

Для того чтобы снять право выполнения, указывается параметр `-x`:

```
chmod -x script
```

Подробнее о символьном методе вы сможете прочитать в руководстве по команде `chmod` (выполнив команду `man chmod`).

14.5.2. Команда *chown*: смена владельца файла

Если вы хотите "подарить" кому-то файл, то есть сделать какого-то пользователя владельцем файла, нужно выполнить команду `chown`:

```
chown пользователь файл
```

> ***Примечание***
>
> Возможно, что после изменения владельца файла вы сами не сможете получить к нему доступ, ведь владельцем будете уже не вы.

14.5.3. Специальные права доступа (SUID и SGID)

Мы рассмотрели обычные права доступа к файлам, но в UNIX есть еще так называемые *специальные права доступа*: SUID (Set User ID root) и SGID (Set Group ID root).

Эти права доступа позволяют обычным пользователям запускать программы, требующие для своего запуска привилегий пользователя root. Например, демон pppd требует привилегий root, но чтобы каждый раз при установке PPP-соединения

(модемное, ADSL-соединение) не входить в систему под именем root, достаточно установить специальные права доступа для демона pppd. Делается это так:

```
chmod u+s /путь/pppd
```

Однако не нужно увлекаться такими решениями, поскольку каждая программа, для которой установлен бит SUID, является потенциальной "дырой" в безопасности вашей системы. Для выполнения программ, требующих прав root, намного рациональнее использовать программы `sudo` и `su` (описание которых можно получить по командам `man sudo` и `man su`).

14.5.4. Команда *chattr*: атрибуты файла, запрет изменения файла

С помощью команды `chattr` можно изменить атрибуты файла. Параметр `+` устанавливает атрибут, а параметр `-` атрибут снимает. Например:

```
# chattr +i /boot/grub/menu.lst
```

Данная команда устанавливает атрибут `i`, запрещающий любое изменение, переименование и удаление файла. Установить этот атрибут, равно как и снять его, имеет право только суперпользователь. Чтобы изменить файл, нужно очистить атрибут с помощью команды:

```
# chattr -i /boot/grub/menu.lst
```

14.6. Монтирование файловых систем

14.6.1. Команды *mount* и *umount*

Чтобы работать с какой-либо файловой системой, необходимо *примонтировать* ее к корневой файловой системе. Например, вставив в дисковод дискету, нужно подмонтировать файловую систему дискеты к корневой файловой системе — только так мы сможем получить доступ к файлам и каталогам, которые на этой дискете записаны. Аналогичная ситуация с жесткими, оптическими дисками и другими носителями данных.

Если вы хотите заменить сменный носитель данных (дискету, компакт-диск), вам нужно сначала размонтировать файловую систему, затем извлечь носитель данных, установить новый и заново смонтировать файловую систему. В случае с дискетой о размонтировании должны помнить вы сами, поскольку при этом выполняется синхронизация буферов ввода/вывода и файловой системы, то есть данные физически записываются на диск, если это еще не было сделано. А компакт-диск система не разрешит вам извлечь, если он не размонтирован. В свою очередь, размонтировать файловую систему можно, только когда ни один процесс ее не использует.

При завершении работы системы (перезагрузке, выключении компьютера) размонтирование всех файловых систем выполняется автоматически.

Команда монтирования (ее нужно выполнять с привилегиями root) выглядит так:

```
# mount [опции] <устройство> <точка монтирования>
```

Точка монтирования — это каталог, через который будет осуществляться доступ к монтируемой файловой системе. Например, если вы подмонтировали компакт-диск к каталогу /mnt/cdrom, то получить доступ к файлам и каталогам, записанным на компакт-диске, можно будет через точку монтирования (именно этот каталог /mnt/cdrom). Точкой монтирования может быть любой каталог корневой файловой системы, хоть /aaa-111. Главное, чтобы этот каталог существовал на момент монтирования файловой системы.

Для монтирования нужно обладать правами root. Можно также воспользоваться командой `sudo`, если пользователь root разрешил вам ее использовать и разрешил монтировать диски (то есть выполнять команду `sudo mount`).

Для размонтирования файловой системы используется команда `umount`:

```
# umount <устройство или точка монтирования>
```

14.6.2. Файлы устройств и их монтирование

В этой главе мы уже говорили о файлах устройств. Здесь мы вернемся к ним снова, но в контексте монтирования файловой системы.

Как уже было отмечено, для BSD нет разницы между устройством и файлом. Все устройства системы представлены в корневой файловой системе как обычные файлы. Например, /dev/fd0 — это ваш дисковод для гибких дисков, /dev/ad*X* (/dev/da*X*) — жесткий диск. Файлы устройств хранятся в каталоге /dev.

Монтирование UFS-разделов

Для монтирования UFS-раздела (например, вы подключили второй жесткий диск от вашего второго вышедшего из строя сервера и пытаетесь добраться до данных, хранящихся на этом жестком диске) нужно ввести команду:

```
# mount [устройство] /mnt
```

Например:

```
# mount /dev/ad1s1a /mnt
```

Имя точки монтирования может быть другим, например, /mnt/ad1s1a (главное, чтобы этот каталог существовал):

```
# mount /dev/ad1s1a /mnt/ad1s1a
```

Аналогично можно подмонтировать другие разделы (d, e, f).

Если при монтировании вы увидели сообщение об ошибке, проверьте этот раздел с помощью утилиты `fsck` (название утилиты происходит от сокращения filesystem check):

```
# fsck /dev/ad1s1a
```

На момент запуска `fsck` раздел не должен быть смонтирован. После устранения ошибок рекомендуется при монтировании раздела использовать опцию `-f`, чтобы пропустить некоторые предупреждения и мелкие ошибки, которые, возможно, не были исправлены:

```
# mount -f /dev/ad1s1a /mnt/ad1s1a
```

По окончании работы с разделом размонтируйте его командой `umount`:

```
# umount [точка монтирования]
```

Например:

```
# umount /dev/ad1s1a
```

Чтобы размонтировать все файловые системы, кроме корневой, нужно ввести команду:

```
# umount -a
```

А чтобы потом смонтировать все файловые системы, указанные в файле /etc/fstab (см. далее), нужно использовать обратную команду:

```
# mount -a
```

Монтирование Windows-раздела (VFAT)

Представим, что у нас есть SATA-винчестер, подключенный к первому SATA-контроллеру. Как мы уже выяснили ранее, он будет называться /dev/ad4. Пусть на этом винчестере организованы два слайса: ad4s1, в котором установлена Windows, и ad4s2, где установлена FreeBSD.

Смонтировать Windows-раздел к каталогу /mnt можно так:

```
# mount_msdosfs /dev/ad4s1 /mnt
```

Можно использовать и команду `mount`, но тогда нужно уточнить тип файловой системы с помощью параметра `-t`:

```
# mount -t msdosfs /dev/ad4s1 /mnt
```

> ***Примечание***
>
> Обязательно указывайте тип файловой системы, иначе при попытке монтирования получите сообщение об ошибке: **Invalid argument**.

После этого обратиться к файлам Windows-раздела можно через каталог /mnt:

```
# ls /mnt
```

Размонтировать раздел можно так:

```
# umount /mnt
```

или

```
# umount /dev/ad4s1
```

Если при монтировании Windows-раздела вы получаете сообщение о недопустимом типе файловой системы, вам нужно добавить в файл /boot/loader.conf строку:

```
msdosfs="YES"
```

После этого нужно сохранить файл и перезагрузить машину.

Монтирование NTFS-раздела (слайса)

А вот если наш Windows-раздел отформатирован как NTFS, нужно установить драйвер fuse-ntfs (так же известный как NTFS-3G), поддерживающий запись на NTFS-раздел:

```
# cd /usr/ports/sysutils/fusefs-ntfs/
# make install clean
```

После установки драйвера добавьте следующую строку в файл /etc/rc.conf:

```
fusefs_enable="YES"
```

Запустим демон fusefs:

```
# /usr/local/etc/rc.d/fusefs start
```

Правильно монтировать NTFS-раздел нужно с указанием опций `rw` (поддержка чтения/записи) и `locale` (задает применение кодировки UTF8, которая используется на NTFS-разделах):

```
# ntfs-3g -o rw,locale=ru_RU.UTF-8 /dev/ad4s1 /mnt
```

Можно также использовать команду `mount_ntfs-3g` — как кому больше нравится:

```
# mount_ntfs-3g /dev/ad4s1 /mnt
```

У модуля ntfs-3g есть только один недостаток — он выводит информацию в кодировке UTF-8. Установить кодировку KOI8-R почему-то не получается. Поэтому если у вас локаль KOI8-R, то имена файлов с русскими буквами будут отображаться неправильно. Один из вариантов — полностью переход на UTF-8 (чтобы локаль тоже была в кодировке UTF-8), но данный вариант подойдет вам, только если вы часто работаете с NTFS-разделами.

Монтирование флешки

Поскольку у нас винчестер SATA и нет SCSI-винчестеров, то у первой подключенной флешки будет имя устройства /dev/da0. Чтобы точно узнать, какое имя присвоено вашей флешке, достаточно ввести команду:

```
# tail /var/log/messages
```

Среди всего прочего в выводе команды вы найдете примерно такие строки:

Nov 14 14:24:22 denhost kernel: da0 at umass-sim0 bus 0 target 0 lun 0

Nov 14 14:24:22 denhost kernel: da0: <производитель, модель> Removable Direct Access SCSI-2 device

Nov 14 14:24:22 denhost kernel: da0: 40.000MB/s transfers

В данном случае флешка, подключенная к компьютеру, получила имя /dev/da0. Для ее монтирования нужно воспользоваться командой:

```
# mount –t msdosfs /dev/da0s1 /mnt
```

Эта команда практически будет универсальной, и ее не нужно изменять. Дело в том, что *обычно* файловая система флешки — VFAT, поэтому тип `msdosfs` подойдет при ее монтировании. И *обычно* вы подключаете одну флешку, а учитывая отсутствие SCSI-винчестеров в большинстве случаев, получаем для нее имя /dev/da0s1, поскольку на флешке *обычно* и есть только один раздел (слайс). Даже если вы подключите вторую флешку, то вам нужно будет изменить только номер устройства — /dev/da1s1 и точку монтирования. Каталог /mnt также в большинстве случаев имеется. Хотя рекомендуется создать каталог /mnt/usb и монтировать флешки к этому каталогу.

> ***Примечание***
>
> В предыдущем абзаце я специально выделил слово *обычно*. Да, описан стандартный случай. Однако у всех разные конфигурации и разные задачи: на вашей машине может стоять SCSI-винчестер, или вы можете подключить три флешки сразу. Тогда эта команда, естественно, работать не будет (точнее, подойдет только для монтирования первой флешки).

Вместо команды `mount` можно использовать команду `mount_msdosfs`. Преимущество этой команды заключается в том, что можно указать кодировку, в которую нужно перекодировать имена файлов на FAT-разделе:

```
# mount_msdosfs -L ru_RU.KOI8-R /dev/da0s1 /mnt
```

Если вы перешли на локаль UTF-8, то вам нужно использовать значение `ru_RU.UTF-8` для параметра `-L`.

Флешка — это сменный и часто используемый носитель, поэтому имеет смысл разрешить монтирование флешек обычным пользователям (чтобы никому не предоставлять лишних полномочий). Для этого откройте файл /etc/sysctl.conf и измените значение параметра vfs.usermount — его нужно установить в 1:

```
vfs.usermount=1
```

После этого откройте файл /etc/devfs.rules (или создайте его, если он не существует) и добавьте в него строки:

```
[localrules=10]
        add path 'da*0*' mode 0660 group operator
```

В файл /etc/rc.conf добавьте строку:

```
devfs_system_ruleset="localrules"
```

После перезагрузки обычные пользователи смогут монтировать свои флешки. Если на флешке есть файлы с русскими буквами в именах, следует указать параметры локализации при монтировании:

```
$ mount -t msdosfs -o -L=ru_RU.KOI8-R /dev/da0s1 ~/usb
```

Монтирование CD/DVD

Монтирование CD/DVD осуществляется очень просто — вам не нужно указывать ни номер устройства, ни точку монтирования. Просто введите команду (перед этим вставьте диск в привод):

```
# mount /cdrom
```

Такая простота достигается благодаря следующей строке в файле /etc/fstab:

```
/dev/acd0               /cdrom          cd9660 ro,noauto      0       0
```

Если у вас два привода CD/DVD и вы хотите также просто монтировать второй привод, тогда добавьте в файл fstab строку (возможно, вам придется изменить имя устройства — acd1):

```
/dev/acd1               /cdrom1         cd9660 ro,noauto      0       0
```

После этого создайте каталог /cdrom1:

```
# mkdir /cdrom1
```

Введите команду: `# mount -a`

После этого для монтирования второго привода достаточно будет ввести команду:

```
# mount /cdrom1
```

Монтирование файловых систем Ext2/3/4, ReiserFS, ZFS

Для монтирования файловой системы Linux нужно сначала загрузить модуль ядра (ext2fs или reiserfs — зависит от типа файловой системы):

```
# kldload ext2fs
# kldload reiserfs
```

Модуль ext2fs обеспечивает поддержку файловых систем ext2, ext3 и ext4.

После этого смонтировать файловую систему можно командами (имя устройства у вас может быть другим):

```
# mount -t ext2fs /dev/ad1s1 /mnt
# mount -t reiserfs /dev/ad1s1 /mnt
```

В Linux раздел (слайс) не делится на разделы (a, b, e и т. д.), поэтому после номера слайса ничего указывать не нужно.

Проверить файловую систему ext2 можно командой `fsck.ext2`, но программу fsck.ext2 придется установить из портов:

```
# cd /usr/ports/sysutils/e2fsprogs
# make install clean
```

> ***Примечание***
>
> Ранее порт, содержащий утилиту fsck.ext2, назывался fsck_ext2fs, но в последних версиях портов он перемещен в порт sysutils/e2fsprogs, что можно узнать по команде `make search name="fsck"`, выполненной из каталога /usr/ports.

Можно также установить пакет e2fsprogs (так будет быстрее, чем установка из портов):

```
# pkg_add -r e2fsprogs
```

Для проверки файловых систем ext3 и ext4 служат утилиты fsck.ext3 и fsck4.

Чтобы каждый раз при загрузке не вводить команду `kldload ext2fs`, добавьте в файл /boot/loader.conf строку:

```
ext2fs_load="YES"
```

Аналогично для ReiserFs нужно добавить строку:

```
reiserfs_load="YES"
```

а для ZFS — строку:

```
zfs_load="YES"
```

Если модуль ext2fs вообще не собран, то смонтировать файловую систему Linux не получится. В этом случае его придется собрать самостоятельно:

```
# cd /usr/src/sys/modules/ext2fs
# make
# make install
```

После этого нужно перезагрузить компьютер. Можно также перекомпилировать ядро *(см. главу 21)*, добавив в конфигурационный файл нового ядра строку:

```
options "EXT2FS"
```

Эта строка обеспечит поддержку файловой системы Linux на уровне ядра без использования модулей.

Во избежание потери данных рекомендуется монтировать "чужие" файловые системы в режиме "только чтение", добавив опцию `-o ro`:

```
# mount -t reiserfs -o ro /dev/ad1s1 /mnt
```

14.6.3. Монтирование разделов при загрузке

Если вы не хотите при каждой загрузке монтировать постоянные файловые системы, нужно прописать их в файле /etc/fstab. Формат файла fstab следующий:

```
устройство точка_монтирования тип_ФС опции флаг_РК флаг_проверки
```

Здесь: `тип_ФС` — это тип файловой системы, а `флаг_РК` — флаг резервного копирования (`dump`). Если он установлен (`1`), то программа `dump` заархивирует данную файловую систему при создании резервной копии. Если не установлен (`0`), то резервная копия этой файловой системы создаваться не будет. `Флаг проверки` устанавливает, будет ли данная файловая система проверяться на наличие ошибок программой `fsck`. Проверка производится в двух случаях:

- если файловая система размонтирована некорректно;
- если достигнуто максимальное число операций монтирования для этой файловой системы.

Поле опций содержит важные параметры файловой системы. Некоторые из них представлены в табл. 14.5.

Таблица 14.5. *Опции монтирования файловой системы в файле /etc/fstab*

Опция	Описание
`auto`	Файловая система должна монтироваться автоматически при загрузке. Опция используется по умолчанию, поэтому ее указывать не обязательно
`noauto`	Файловая система не монтируется при загрузке системы (при выполнении команды `mount -a`), но ее можно смонтировать вручную с помощью все той же команды `mount`
`async`	Асинхронный режим ввода/вывода
`noatime`	Запрещает изменение атрибута времени последнего доступа к файлу, что ускоряет работу файловой системы
`ro`	Монтирование в режиме "только чтение"
`rw`	Монтирование в режиме "чтение/запись". Используется по умолчанию для файловых систем, поддерживающих запись

Примечание

Редактировать файл /etc/fstab, как и любой другой файл из каталога /etc, можно в любом текстовом редакторе (например, ee, mcedit, vi), но перед этим нужно получить права root.

Рассмотрим небольшой пример:

```
# Device            Mountpoint    FStype  Options       Dump    Pass#
/dev/ad0s1b         none          swap    sw            0       0
/dev/ad0s1a         /             ufs     rw            1       1
/dev/ad0s1e         /tmp          ufs     rw            2       2
/dev/ad0s1f         /usr          ufs     rw            2       2
/dev/ad0s1d         /var          ufs     rw            2       2
/dev/acd0           /cdrom        cd9660  ro,noauto     0       0
```

Раздел ad0s1b — это раздел подкачки (тип swap), для него не нужно указывать точку монтирования, этот раздел не нужно проверять и копировать программой dump. Раздел ad0s1a используется в качестве корневой файловой системы (точка монтирования /). Остальные разделы монтируются к каталогам /tmp, /usr и /var.

Тип файловой системы — ufs. Для корневой файловой системы флаги dump и pass устанавливаются в 1, для всех остальных — в 2 (если нужен dump и проверка).

Из опций для всех записей (кроме первой и последней) указывается только rw, разрешающая запись и чтение раздела.

Последняя запись используется для неавтоматического монтирования привода CD/DVD. Если у вас нет SCSI-дисков и вы часто используете флешку, тогда можно добавить в /etc/fstab запись:

```
/dev/da0s1     /mnt/usb              msdosfs        rw,noauto      0       0
```

После этого для монтирования флешки можно будет использовать команду:

```
# mount /mnt/usb
```

Только не забудьте создать каталог /mnt/usb:

```
# mount /mnt/usb
```

Для каталога /tmp можно смело указать опции noatime и async — это повысит производительность файловой системы для временных файлов:

```
/dev/ad0s1e            /tmp            ufs     rw,noatime,async      2       2
```

Если время последнего доступа к файлу вам знать не нужно, можно добавить опцию noatime для каждой файловой системы (с типом ufs) — это несколько повысит производительность.

Как уже было указано, смонтировать все файловые системы, указанные в /etc/fstab (для которых не указана опция noauto), можно командой:

```
# mount -a
```

Если требуется автоматически смонтировать только файловые системы определенного типа, следует указать параметр -t:

```
# mount -a -t ufs
```

14.7. Полезные примеры

14.7.1. Монтирование ISO-образа

В Windows для монтирования (читайте — для просмотра содержимого) ISO-образа нужно установить дополнительную программу, а вот в FreeBSD можно использовать средства файловой системы. Для монтирования ISO-образа FreeBSD.iso к каталогу /mnt введите следующие две команды:

```
# mdconfig -a -t vnode -u0 -f FreeBSD.iso
# mount -t cd9660 /dev/md0 /mnt
```

Размонтирование образа осуществляется следующей командой:

```
# umount /dev/md0; mdconfig -d -u0
```

Иногда ядро бывает собрано без поддержки md, поэтому для монтирования образа придется использовать другой способ:

```
# vnconfig /dev/vn0c FreeBSD.iso
# mount -t cd9660 /dev/vn0c /mnt
```

Размонтировать образ можно командой:

```
# umount /mnt; vnconfig -u /dev/vn0c
```

14.7.2. Монтирование каталога

В UNIX можно подмонтировать один каталог к другому. Зачем это вам нужно, вы должны знать сами (иногда проще использовать ссылки). Для монтирования каталога А к каталогу Б используется команда:

```
# mount_null А Б
```

14.7.3. Проблемы со SCSI-приводами DVD

В некоторых случаях SCSI-привод не успевает ответить на сброс шины, поэтому при монтировании DVD вы можете увидеть сообщение **Device not configured** или же вообще в системе не будет видно устройства /dev/cd*N*. Хорошего мало — придется перекомпилировать ядро *(см. главу 21)*, добавив в файл конфигурации ядра строку:

```
options SCSI_DELAY=15000
```

Ну, а если уж 15 секунд не хватит на ответ, скорее всего, привод DVD придется заменить.

14.8. Добавление еще одного жесткого диска

Предположим, что нам одного жесткого диска мало, и мы прикупили второй жесткий диск, который планируем отформатировать в UFS и использовать в BSD. При установке системы была запущена программа fdisk с минимальным интерфейсом пользователя, с которым вы уже знакомы и который, если к нему привыкнуть, довольно удобен. Но при вводе команды `fdisk <имя диска>` мы получаем просто список разделов на жестком диске — немного не то, чего мы ожидали.

Проще всего добраться к тому самому интерфейсу через программу sysinstall. Запустите sysinstall, выберите команду **Configure | Fdisk** (рис. 14.6). Затем выберите ваш второй диск — в моем случае это ad1 (рис. 14.7).

Наконец, вы увидите тот самый интерфейс разметки диска (рис. 14.8). Нажмите клавишу <C> для создания слайса вручную. Далее нужно указать размер слайса в блоках (по умолчанию создается раздел на весь жесткий диск) или в мегабайтах, добавив букву M к числу (например, 100M). Нас интересует один большой раздел, потому просто нажимаем **OK** (рис. 14.9).

В следующем окне укажите номер типа слайса (для FreeBSD это номер 165) и нажмите **OK** (рис. 14.10).

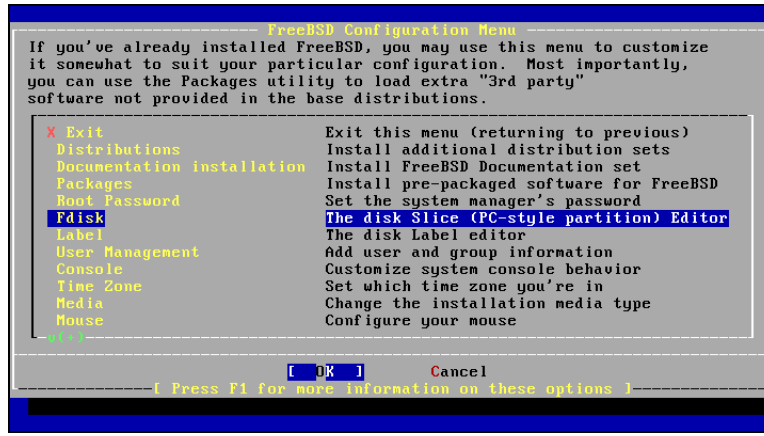


Рис. 14.6. Конфигуратор sysinstall

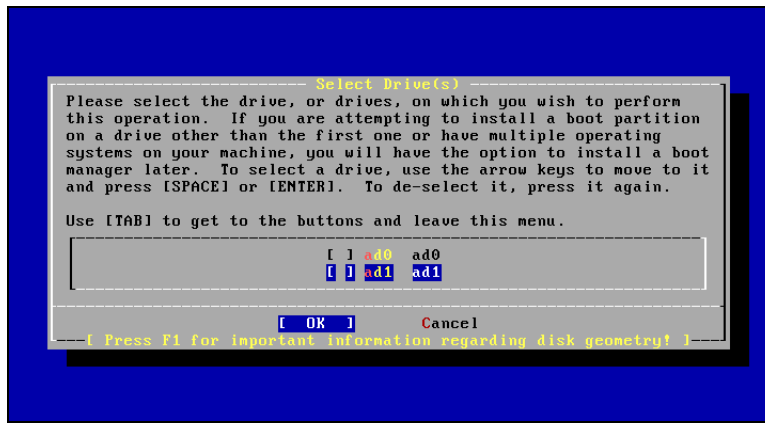


Рис. 14.7. Выбор диска для изменения разделов

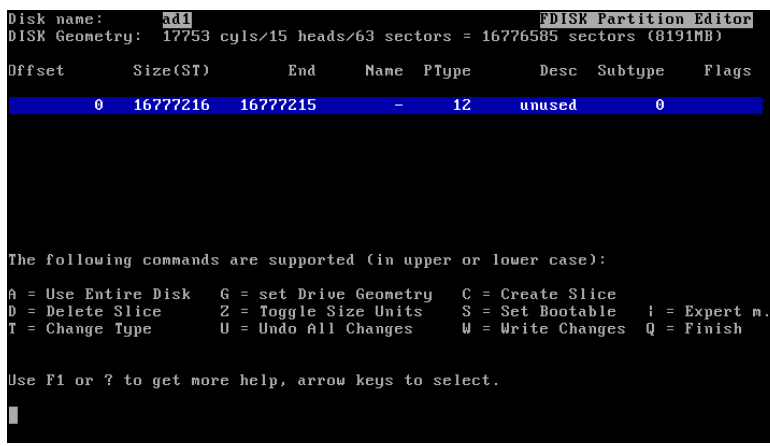


Рис. 14.8. Разметка диска отсутствует

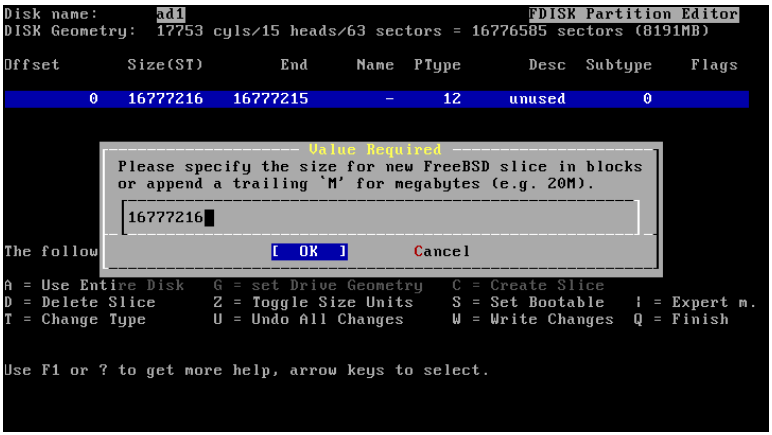


Рис. 14.9. Размер нового слайса

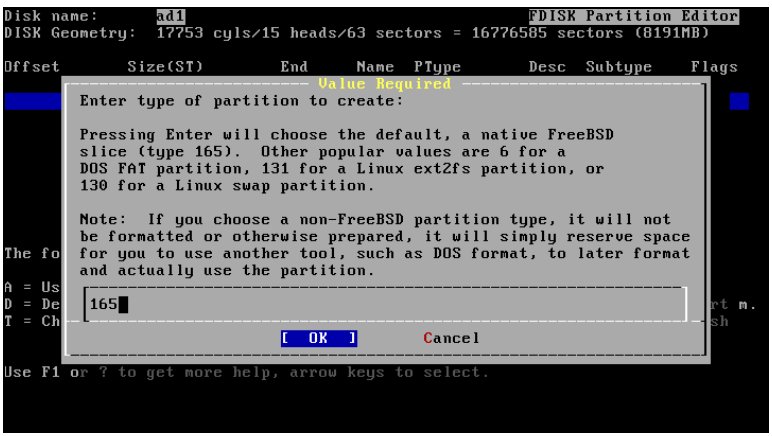


Рис. 14.10. Номер типа слайса

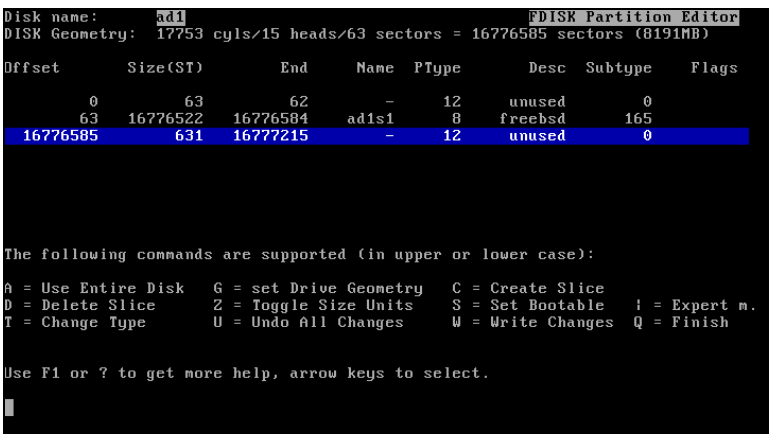


Рис. 14.11. Разметка диска завершена

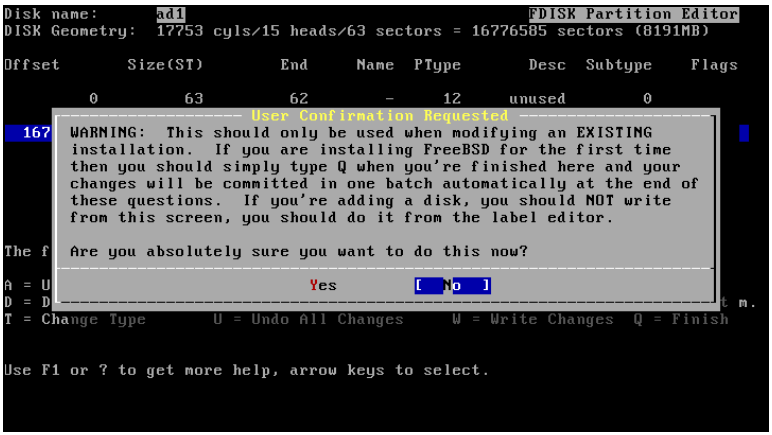


Рис. 14.12. Выберите **Yes**

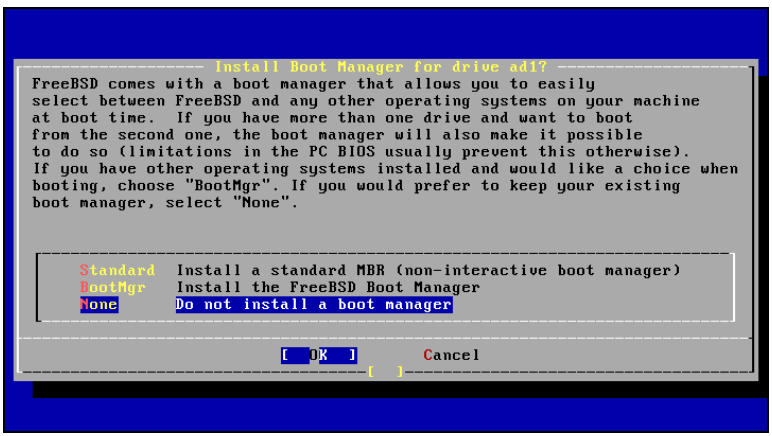


Рис. 14.13. Нам не нужно устанавливать загрузчик

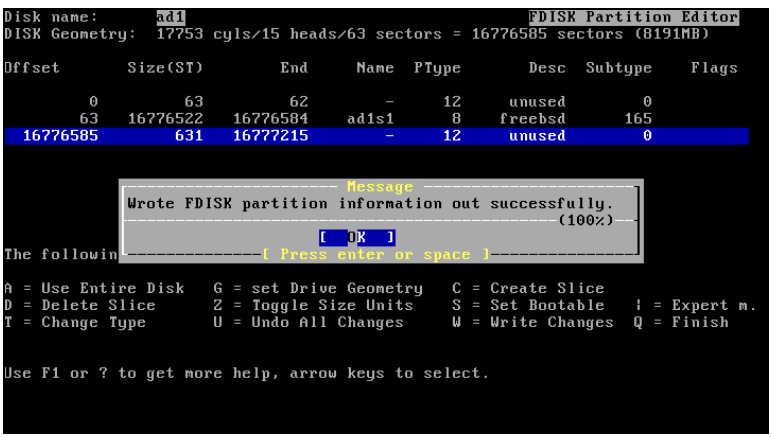


Рис. 14.14. Разметка диска успешно завершена



Рис. 14.15. BSD-разделы не созданы

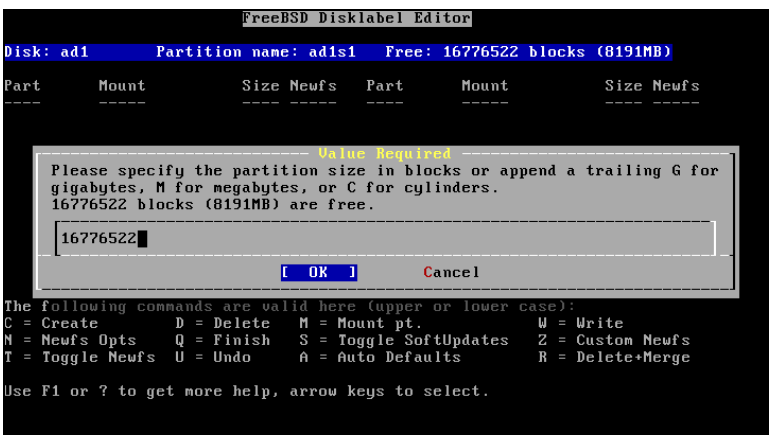


Рис. 14.16. Ввод размера BSD-раздела

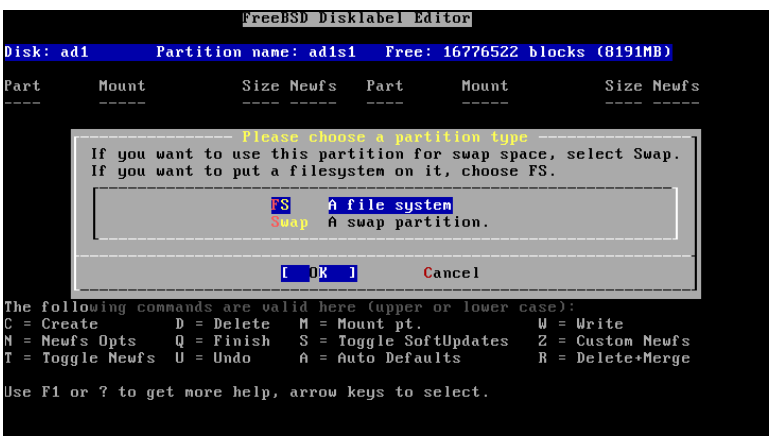


Рис. 14.17. Тип BSD-раздела

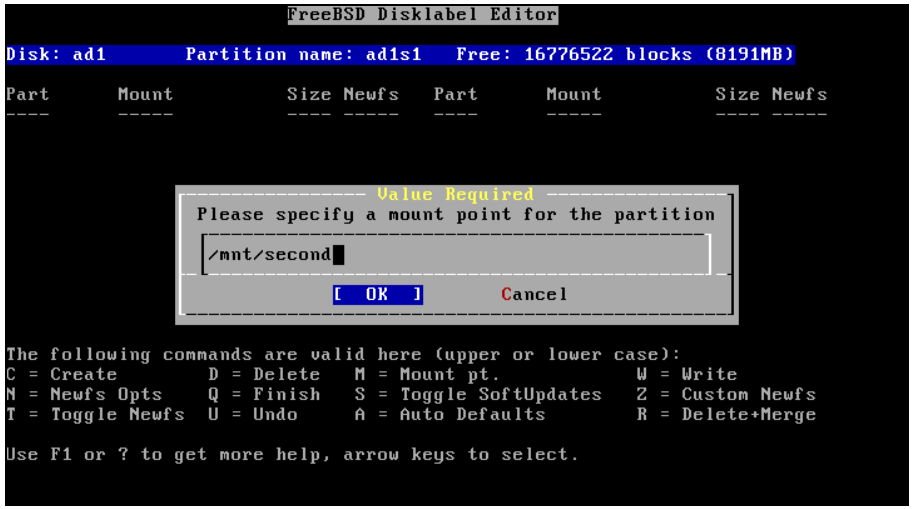


Рис. 14.18. Ввод точки монтирования

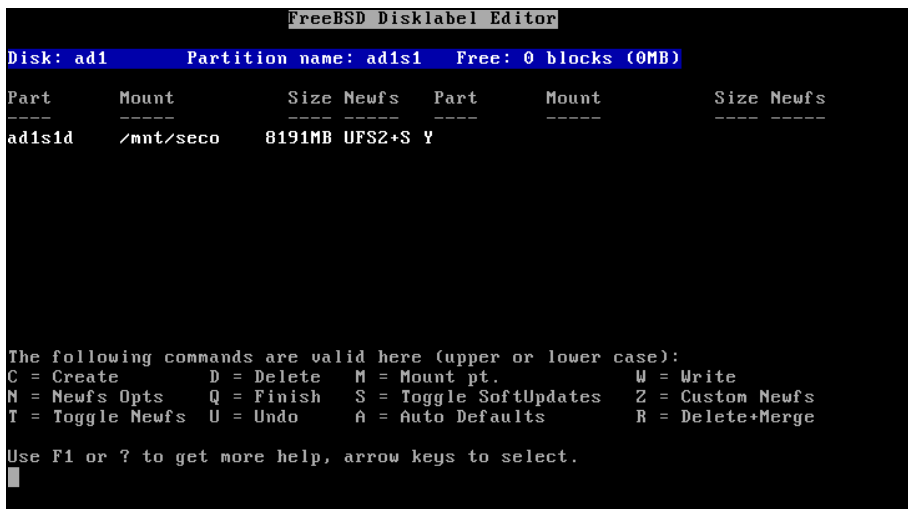


Рис. 14.19. Разметка слайса

```
denhost# bsdlabel /dev/ad1s1
# /dev/ad1s1:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  c: 16776522        0    unused        0     0         # "raw" part, don't edit

  d: 16776522        0    4.2BSD        0     0     0
denhost# mount /dev/ad1s1d /mnt/second
denhost# ls /mnt/second
.snap
denhost#
```

Рис. 14.20. Просмотр разметки BSD-слайса

В результате получится разметка диска, приведенная на рис. 14.11, — всего один раздел. Нажмите клавишу <W> для записи изменений на диск. В открывшемся окне с ужасным предупреждением нажмите **Yes** (рис. 14.12).

Затем вам будет предложено установить загрузчик на новый жесткий диск — отказываемся от этого, выбрав **None** и нажав **OK** (рис. 14.13).

Теперь вас поздравят с успешной разметкой диска (рис. 14.14).

Нажав **OK**, вы вернетесь в окно fdisk. Нажмите клавишу <Q> для выхода, и вы вернетесь в окно выбора диска (см. рис. 14.7). На этот раз выберите **Cancel**, и вы вернетесь в меню настройки sysinstall (см. рис. 14.6). Теперь нужно выбрать опцию **Label** для запуска редакторов BSD-разделов. По умолчанию BSD-разделы не созданы (рис. 14.15) — было бы удивительно, если бы fdisk создавал еще и BSD-разделы.

Поскольку нам не нужны точки монтирования для /usr, /tmp, не нужен раздел подкачки, то автоматическую разметку командой **A** нам использовать противопоказано. Опять нажимаем клавишу <C> для создания на этот раз уже BSD-раздела, после чего откроется окно ввода размера (рис. 14.16). Можете создать один большой раздел — все зависит от того, какую конфигурацию вы планируете.

Следующий этап (рис. 14.17) — выбор типа BSD-раздела: **FS** (файловая система) или **Swap** (подкачка).

Теперь надо назначить для этого раздела точку монтирования (рис. 14.18). Я ввел точку монтирования `/mnt/second`. Данный каталог должен существовать на момент запуска sysinstall. Если вы его не создали, укажите просто `/mnt` — потом исправите.

В результате была создана конфигурация, изображенная на рис. 14.19, — на диске появился раздел ad1s1d емкостью в 8 Гбайт. Новый раздел получил имя d, поскольку имена a, b, c зарезервированы.

Для выхода нажмите клавишу <Q>, затем выйдите из sysinstall. Проверить созданную метку можно командой:

```
# bsdlabel /dev/ad1s1
```

После этого мы подмонтировали раздел /dev/ad1s1d к каталогу /mnt/second (рис. 14.20).

Что делать с этим диском дальше? Если у вас большая база данных или огромный Web-сервер, файлы которых не помещаются на первом жестком диске, можно перенести их на второй жесткий диск:

```
# cp -R /var/* /mnt/second
```

После этого измените точку монтирования /var в fstab:

```
/dev/ad1s1d          /var          ufs     rw            2       2
```

Далее можно выполнить команды `umount -a`, а затем `mount -a`, чтобы не перезагружать компьютер. Впрочем, учитывая, что MySQL-сервер и Apache запущены, компьютер лучше всего, все-таки, перезагрузить. Когда компьютер загрузится снова, то данные Web-сервера (и всех прочих сервисов, что хранят данные в каталоге /var) будут уже на диске ad1, а не на ad0.

14.9. Еще раз о Midnight Commander

Со времен DOS мы знакомы с классическим двухпанельным файловым менеджером Norton Commander. В FreeBSD можно установить файловый менеджер Midnight Commander, который как две капли воды похож на NC (рис. 14.21). Для установки mc (если вы этого не сделали в *главе 8*) введите команды:

```
# pkg_add -r mc
# rehash
# mc
```

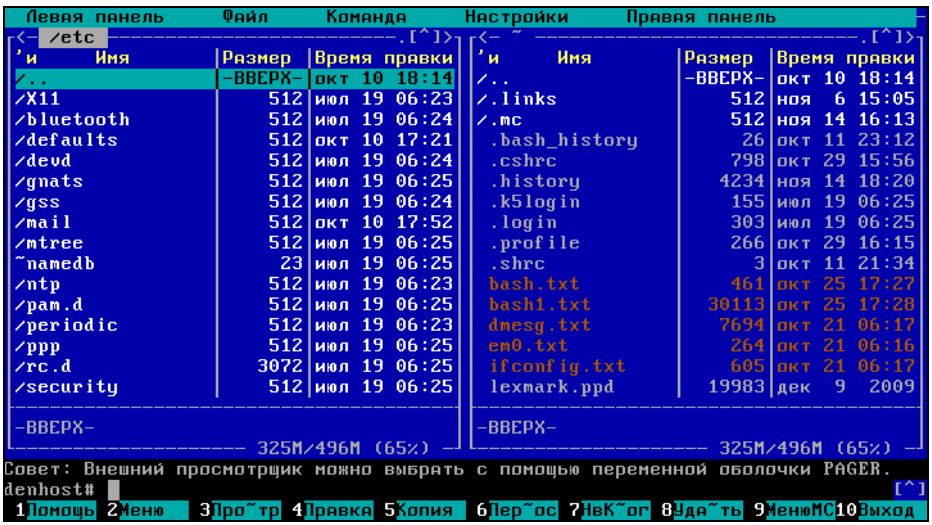


Рис. 14.21. Файловый менеджер Midnight Commander

С помощью mc намного удобнее выполнять рутинные операции: копирование, переименование, архивирование, поиск файлов. К тому же mc оснащен встроенным и очень удобным текстовым редактором, который можно запускать отдельно — командой `mcedit`. Может, использование mc это и не в стиле UNIX-гуру, но зато удобно.

Если mc долго запускается, значит, у вас есть проблема с DNS. Для решения этой проблемы проще всего имя вашего узла указать в файле /etc/hosts:

```
127.0.0.1       <имя узла>
```

На медленных терминалах (когда вы получаете удаленный доступ к системе по медленному каналу через SSH) можно использовать опцию `-s`, которая сокращает вывод лишних символов (mc будет выглядеть не так привлекательно, зато станет работать быстрее): `mc -s`



Глава 15

Пользователи и группы. Квотирование

15.1. Многопользовательская система

BSD — многозадачная многопользовательская операционная система. Это означает, что в один момент с системой могут работать несколько пользователей, и каждый пользователь может запустить несколько приложений. При этом вы можете зайти в систему локально, а кто-то — удаленно, используя один из протоколов удаленного доступа (telnet, SSH) или по FTP. Согласитесь, очень удобно. Предположим, вы забыли распечатать очень важный документ, а возвращаться домой уже нет времени. Если ваш компьютер должным образом настроен и подключен к Интернету, вы можете получить к нему доступ (даже если компьютер выключен, достаточно позвонить домой и попросить кого-то включить его, а к Интернету компьютер подключится автоматически). После чего зайдите в систему по SSH (или подключитесь к графическому интерфейсу, если вы предпочитаете работать в графическом режиме) и скопируйте нужный вам файл. Даже если кто-то в момент вашего подключения уже работает с системой, вы не будете мешать друг другу.

Вы можете обвинить меня в рекламе BSD: мол, эта возможность была и в Windows 98, если установить соответствующее программное обеспечение вроде Remote Administrator. Должен отметить, что в Windows все иначе. Да, Remote Administrator предоставляет удаленный доступ к рабочему столу, но если за компьютером уже работает пользователь, то вместе вы работать не сможете — вы будете мешать ему, а он вам. Ведь все, что будете делать вы, будет видеть он, а все, что будет делать он, вы увидите у себя на экране, то есть рабочий стол получится как бы общий. Если вы предварительно не предупредите пользователя о своем удаленном входе, он даже может подумать, что с системой что-то не то. Помню, со мной так и было — пользователь, работавший за компьютером, закрывал окна, которые я открывал, работая в удаленном режиме. Пришлось мне самому пойти к компьютеру того пользователя и попросить его не мешать.

В BSD же все так, как и должно быть. Несколько пользователей могут работать с системой и даже не подозревать о существовании друг друга, пока не введут соответствующую команду (`who`).

15.2. Пользователь root

15.2.1. Максимальные полномочия

Пользователь root обладает в системе максимальными полномочиями. Система полностью подвластна этому пользователю, и любая его команда будет системой безоговорочно выполнена. Поэтому работать под именем пользователя root нужно с осторожностью. Всегда думайте над тем, что собираетесь сделать. Система выполнит даже команду на удаление корневой файловой системы. Если же вы попытаетесь выполнить определенную команду, зарегистрировавшись под именем обычного пользователя, система сообщит вам, что у вас нет полномочий.

Представим, что кто-то решил пошутить и выложил в Интернете (записал на диск или прислал по электронной почте — не важно) вредоносную программу. Если вы ее запустите от имени пользователя root, система может быть уничтожена. Запуск этой же программы от имени обычного пользователя ничего страшного не произведет — система просто откажется ее выполнять.

Или все будет намного проще — вы ошибочно введете команду, которая может разрушить вашу систему. Или просто отойдете ненадолго от своего компьютера, а тут сразу же появится доброжелатель, — имея полномочия пользователя root, уничтожить систему можно одной командой. Именно поэтому во многих дистрибутивах Linux вход как root или запрещен или ограничен (например, вход разрешен только в консоли, но не в графическом режиме). Но в BSD считается, что вы знаете, что делаете, поэтому вход как root никак не ограничивается.

Можно сделать следующие выводы:

- старайтесь реже работать пользователем root;
- всегда думайте, какие программы вы запускаете под именем root;
- если вы не знаете, что делает та или иная команда (например, скачали ее с какого-то сайта), не вводите ее, пока не выясните, что она делает (man всегда под рукой);
- если программа, полученная из постороннего источника, требует root-полномочий, это должно насторожить;
- создайте обычного пользователя (даже если вы сами являетесь единственным пользователем компьютера) и рутинные операции (с документами, использование Интернета и т. д.) производите от имени этого пользователя.

Если полномочия root все же нужны, совсем необязательно заходить в систему под этим пользователем, достаточно запустить терминал и выполнить команду `sudo` или `su`. После этого в терминале можно выполнять команды с правами root. Если вы закроете терминал, то больше не сможете работать с правами root. Очень удобно — ведь обычно права root нужны для одной-двух операций (например, выполнить команду установки программы или создать/удалить пользователя).

Команда *sudo*

Команда `sudo` позволяет запустить любую команду с привилегиями root. Использовать ее нужно так:

```
sudo <команда_которую_нужно_выполнить_с_правами_root>
```

Например:

```
sudo ee /etc/rc.conf
```

Если ввести эту же команду, но без `sudo` (просто `ee /etc/rc.conf`), текстовый редактор тоже запустится и откроет файл, но сохранить изменения вы не сможете, поскольку у вас не хватит полномочий.

Программа `sudo` перед выполнением указанной вами команды запросит у вас пароль:

```
sudo ee /etc/rc.conf
Password:
```

Вы должны ввести свой *пользовательский пароль* — тот, который применяете для входа в систему, но не пароль пользователя root.

Использовать команду `sudo` имеют право не все пользователи, а только те, которые внесены в список sudoers. Администратор системы (пользователь root) может редактировать этот файл с помощью команды `visudo`.

По умолчанию команда `sudo` не установлена. Для ее установки введите команды:

```
# cd /usr/ports/security/sudo/
# make install clean
# rehash
```

На момент ввода второй команды у вас должны быть установлены порты (это мы сделали при установке системы) и соединение с Интернетом.

После установки команды `sudo` нужно запустить команду `visudo` и внести пользователей, которым разрешено использовать `sudo`, в список sudoers:

```
# visudo
```

Каждая строка в списке sudoers содержит имя пользователя, тип аутентификации и список команд, которые может выполнить пользователь как root. Рассмотрим несколько записей списка sudoers:

```
denis ALL=(ALL) NOPASSWD: ALL
ray   localhost = NOPASSWD: /bin/kill, /bin/ls, /usr/bin/lprm
max   denhost = NOPASSWD: /bin/kill, PASSWD: /bin/ls, /usr/bin/lprm
```

Указанным в списке пользователям разрешается с применением команды `sudo` выполнять определенные списком команды.

- Так, пользователь `denis` может запускать от имени root любые команды на любой машине даже без ввода пароля (опция `NOPASSWD`).

> ***Пояснение***
>
> Опция `NOPASSWD` позволяет выполнять указанные в ней команды без ввода пароля. Например, вводим команду `sudo reboot`. Обычно после этого запрашивается пароль, а если указана опция `NOPASSWD`, пароль запрашиваться не будет.

Выходит, что пользователю `denis` можно доверять как себе. Обычно такую запись делают для пользователя, от имени которого вы работаете каждый день (под root работать не рекомендуется — мы уже знаем почему).

- Пользователю `ray` на машине `localhost` разрешается вводить команды `kill`, `ls`, `lprm` без пароля.
- Пользователю `max` на машине `denhost` без пароля разрешается использовать только команду `kill`, а команды `ls` и `lprm` можно вводить только с паролем.

Примечание

Понятно, что речь идет только о выполнении команд с полномочиями root. Со своими правами пользователь может вводить команды без всякого пароля.

В файле /usr/local/etc/sudoers (в нем и хранится список sudoers) вы найдете еще несколько полезных примеров.

Вам надоело каждый раз вводить `sudo` в начале команд? Тогда выполните команду:

```
sudo -i
```

Эта команда запустит оболочку root. То есть вы сможете вводить любые команды, и они будут выполнены с правами root.

Команда *su*

Команда `su` позволяет получить доступ к консоли root любому пользователю (даже если пользователь не внесен в файл sudoers) при условии, что он знает пароль root. Впрочем, тут все зависит от настроек системы. Эти самые настройки в FreeBSD хранятся в файле /etc/pam.d/su. По умолчанию использовать `su` могут только члены группы wheel. Но в других UNIX-системах настройки по умолчанию таковы, что `su` может выполнить любой пользователь.

Понятно, что в большинстве случаев этим пользователем будет сам пользователь root — не будете же вы всем пользователям доверять пароль root? Поэтому команда `su` предназначена, в первую очередь, для администратора системы, а `sudo` — для остальных пользователей, которым иногда нужны права root (чтобы они меньше отвлекали администратора от своей работы).

Пояснение

Преимущество у `sudo` над `su` следующее — вы можете не только определить, кто имеет право получить root-доступ, но и указать, кто и какие команды может использовать, что невозможно сделать с помощью `su`.

Использовать команду `su` просто: `su`

Затем по запросу надо будет ввести пароль пользователя root, и вы сможете работать в консоли, как обычно. Использовать `su` удобнее, чем `sudo`, потому что вам не нужно вводить `su` перед каждой командой, которая должна быть выполнена с правами root.

Чтобы закрыть сессию `su`, введите команду `exit` или (если вы работаете в графическом режиме) просто закройте окно терминала.

15.3. Создание, удаление и модификация пользователей

15.3.1. Создание пользователя: команда *adduser*

Для добавления нового пользователя служит команда `adduser`. Вам будет задан ряд вопросов, на которые в большинстве случаев можно ответить просто нажатием

клавиши <Enter>. Как минимум, вам нужно будет ввести только логин пользователя. Рассмотрим процесс создания пользователя (рис. 15.1) с моими комментариями:

```
# adduser
```

Логин, используется для входа в систему:

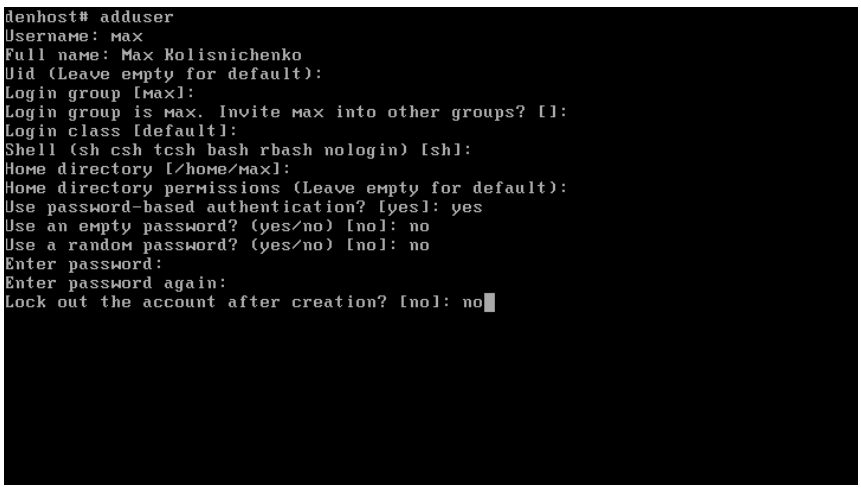
Username: `max`

Полное имя пользователя (можно ничего не вводить):

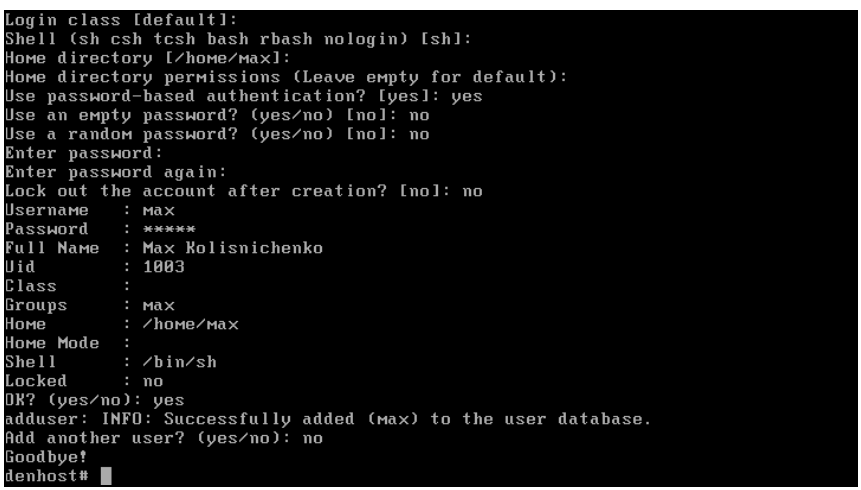
Full name: `Max Kolisnichenko`

Идентификатор (UID) пользователя (можно не вводить, тогда он будет присвоен автоматически):

Uid (Leave empty for default):



а



б

Рис. 15.1. *а* — процесс создания пользователя;
б — *п*роцесс создания пользователя (продолжение)

Основная группа пользователя (можно не вводить, тогда будет создана новая группа с таким же именем, как и у создаваемого пользователя):

Login group [max]:

Можно добавить пользователя в дополнительные группы:

Login group is max. Invite max into other groups? []:

Класс регистрации (можно ничего не вводить, а можно указать `russian`, если консоль пользователя должна быть на русском, — см. *главу 8*):

Login class [default]:

Командный интерпретатор (по умолчанию устанавливается sh, но вы можете использовать другой интерпретатор из списка установленных в вашей системе: csh, tcsh, bash, rbash, nologin):

Shell (sh csh tcsh bash rbash nologin) [sh]:

Домашний каталог пользователя (просто нажмите клавишу <Enter>):

Home directory [/home/max]:

Использовать пароль для входа?

Use password-based authentication? [yes]: `yes`

Использовать пустой пароль?

Use an empty password? (yes/no) [no]: `no`

Сгенерировать случайный пароль?

Use a random password? (yes/no) [yes]: `no`

Введите пароль пользователя (при вводе на экране ничего не отображается):

Enter password:

Повторите пароль:

Enter password again:

Заблокировать учетную запись после создания?

Lock out the account after creation? [no]: `no`

Информация о создаваемом пользователе

Username : max

Password : **

Full Name : Max Kolisnichenko

Uid : 1003

Class :

Groups : max

Home : /home/max)

Shell : /bin/sh

Locked : no

Все ОК?

OK? (yes/no): yes

Пользователь max успешно создан:

adduser: INFO: Successfully added (max) to the user database.

Добавить следующего пользователя?

Add another user? (yes/no): `no`

В реальных условиях вам придется создать не одного, а несколько десятков пользователей. Чтобы не вводить заново некоторые параметры, например, список групп по умолчанию или класс регистрации, можно указать параметры по умолчанию, которые будет предлагать вам команда `adduser`. Для настройки `adduser` запустите ее с параметром `-C`:

```
# adduser -C
```

Любителям делать все одной командой можно посоветовать команду `pw`, эту же команду удобно использовать в сценариях оболочки. Команда `pw` тоже создает пользователя, но делает это не интерактивно (как команда `adduser`). При этом все необходимые параметры указываются в командной строке:

```
# pw useradd denis -d /home/denis -s /bin/tcsh
```

Вот пример создания "помощника" администратора (пользователь помещается в группу wheel):

```
# pw useradd admin -d /home/admin -g wheel -s /bin/tcsh -m -L russian
```

Назначение параметров команды `pw` подробно описано в справочной системе (`man pw`).

15.3.2. Удаление пользователя: команда *rmuser*

Для удаления пользователя служит команда `rmuser`:

```
# rmuser имя_пользователя
```

Как и при создании пользователя программа заставит ответить вас на ряд вопросов:

```
# rmuser max
```

Найден пользователь max:

Matching password entry:

max:*:1003:1003::0:0:Max Kolisnichenko:/home/max:/bin/sh

Удалить?

Is this the entry you wish to remove? `y`

Удалить домашний каталог пользователя?

Remove user's home directory (/home/max)? `y`

Removing user (max): mailspool home passwd.

15.3.3. Изменение пароля пользователя: команда *passwd*

Для изменения пароля служит команда `passwd`. Если запустить ее без параметров, будет инициировано изменение пароля пользователя, запустившего `passwd`. Но можно указать имя пользователя — тогда `passwd` изменит пароль указанного пользователя (ясно, что изменить пароль другого пользователя может только root):

```
# passwd denis
```

15.4. Подробно о создании пользователей

Давайте разберемся, что же происходит при создании новой учетной записи пользователя.

Во-первых, создается запись в файле /etc/passwd. Формат записи следующий:

```
имя_пользователя:пароль:UID:GID:полное_имя:домашний_каталог:оболочка
```

Рассмотрим фрагмент этого файла (две строки):

```
root:x:0:0:root:/root:/bin/csh
den:x:10001:1001:Denis:/home/den:/bin/sh
```

- первое поле — это логин пользователя, который он вводит для регистрации в системе. Пароль в современных системах в этом файле не указывается, а второе поле осталось просто для совместимости со старыми системами. Пароли хранятся в файле /etc/master.passwd, о котором мы поговорим чуть позже;
- третье и четвертое поле — это UID (User ID) и GID (Group ID) — идентификаторы пользователя и группы соответственно. Идентификатор пользователя root всегда равен 0, как и идентификатор группы root. Список групп вы найдете в файле /etc/group;
- пятое поле — это настоящее имя пользователя. Может быть не заполнено, а может содержать фамилию, имя и отчество пользователя — все зависит от педантичности администратора системы, то есть от вас. Если вы работаете за компьютером в гордом одиночестве, то, думаю, свою фамилию не забудете. А вот если ваш компьютер — сервер сети, тогда просто необходимо указать Ф. И. О. каждого пользователя, а то, когда придет время обратиться к пользователю по имени, вы его знать не будете (попробуйте запомнить 500 фамилий и имен!);
- шестое поле содержит имя домашнего каталога. Обычно это каталог /home/<имя_пользователя>;
- последнее поле — это имя командного интерпретатора, который будет обрабатывать введенные вами команды, когда вы зарегистрируетесь в консоли.

В целях безопасности пароли были перенесены в файл /etc/master.passwd (доступен для чтения/записи только пользователю root), где они и хранятся в закодированном виде (используется алгоритм MD5 или Blowfish в некоторых UNIX-системах). Узнать, с помощью какого алгоритма зашифрован пароль, очень просто: посмотрите на шифр — если он достаточно короткий и не начинается с символа $, то применен алгоритм DES (самый слабый и ненадежный — как правило, используется в старых дистрибутивах). Если же шифр начинается с символов 1, то это MD5, а если в начале шифра имеются символы $2a$, то это Blowfish.

Формат файла master.passwd почти такой же, как и файла passwd, но после поля GID добавляются три новых поля: класс регистрации, время смены пароля, время деактивации. Если время смены пароля равно 0, то пользователю никогда не придется изменять пароль. Если время деактивации учетной записи равно 0, то учетная запись всегда будет активной. Время смены пароля и время деактивации указыва-

ется в формате timestamp, то есть задается количество секунд, прошедших от 1 января 1970 года до заданной даты.

Во-вторых, при создании пользователя создается каталог /home/<имя пользователя>, в который копируется содержимое каталога /etc/share/skel. Каталог /etc/share/skel содержит "джентльменский набор" — файлы конфигурации по умолчанию, которые должны быть в любом пользовательском каталоге. Название каталога skel (от skeleton) полностью оправдывает себя — он действительно содержит "скелет" домашнего каталога пользователя.

Для редактирования файла /etc/passwd нельзя использовать обычный текстовый редактор. Для этого предназначена специальная утилита vipw (рис. 15.2). Утилита vipw для редактирования базы пользователей запускает редактор, указанный в переменной окружения `EDITOR`. После редактирования файла (когда вы выходите из редактора с сохранением данных) утилита vipw генерирует специальные хэш-таблицы, которые ускоряют вход пользователя в систему. Ведь если бы база данных о пользователях была в текстовом формате (да, ее приятно редактировать администратору), то процесс получения информации о пользователе при его входе в систему длился бы довольно долго (при наличии нескольких тысяч пользователей). Именно поэтому и были созданы хэш-таблицы, которые формируются на основании файлов /etc/passwd и /etc/master.passwd. Хэш-таблицы называются /etc/pwd.db и /etc/spwd.db. Если вы поспешили и по старинке отредактировали один из файлов (/etc/passwd или /etc/master.passwd) вручную, введите команду `pwd mkdb` для обновления хэш-таблиц.

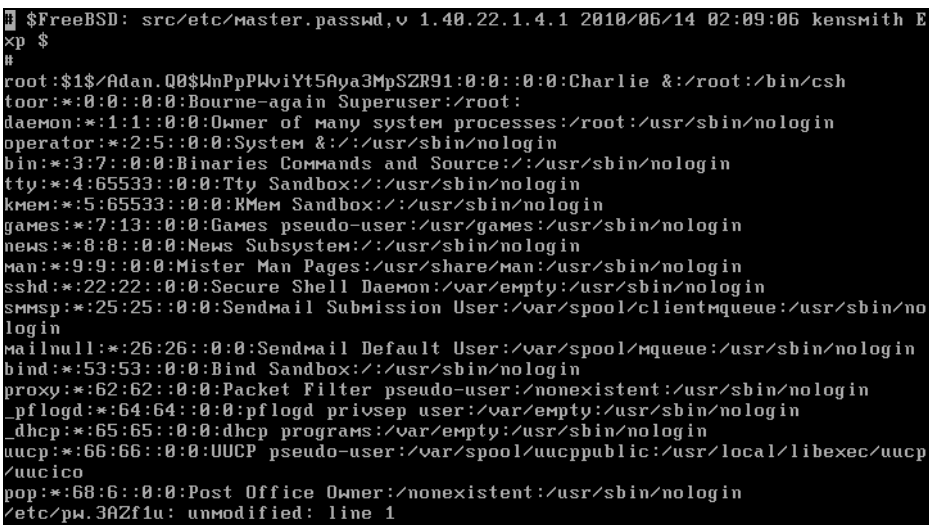


Рис. 15.2. Утилита vipw

15.5. Группы пользователей

Иногда пользователей объединяют в *группы*. Группы позволяют более эффективно управлять правами пользователей. Например, вы можете объединить некото-

рых пользователей в группу admins — все пользователи из этой группы будут обладать особенными правами доступа.

Список групп хранится в файле /etc/group. Для этого файла нет хэш-таблицы, поэтому его можно редактировать в любом текстовом редакторе. Формат файла group прост:

```
имя_группы:пароль_группы:GID:список_пользователей
```

Пароль группы обычно не задается (вместо него указывается символ `*`), GID — это идентификатор группы. Последнее поле — список пользователей (через запятую), входящих в группу. Пример фрагмента файла group:

```
wheel:*:0:root
daemon:*:1:
kmem:*:2:
sys:*:3:
tty:*:4:
operator:*:5:root
admins:x:125:denis,den,max
```

15.6. Ограничение дискового пространства

Механизм ограничения дискового пространства пользователей называется *квотированием*. FreeBSD, как уже отмечалось, система многопользовательская, поэтому без ограничения дискового пространства администратору не обойтись. Когда используешь компьютер в гордом одиночестве, все дисковое пространство доступно вам и только вам. А вот когда пользователей несколько, приходится ограничивать доступное пространство, чтобы один из пользователей не "узурпировал" все место на диске. По какому принципу ограничивать дисковое пространство? Можно поделить его поровну между пользователями, можно одним пользователям отдать больше места, а другим — меньше.

На домашнем компьютере квотирование вряд ли понадобится, а на сервере, как правило, для каталога /home отводится отдельный раздел жесткого диска. Поэтому будем считать, что у нас есть отдельный раздел, который монтируется к каталогу /home.

Для включения квот необходимо соответственным образом перекомпилировать ядро. Для этого в файл конфигурации ядра *(см. главу 21)* нужно добавить строку:

```
options QUOTA
```

После перекомпиляции ядра добавьте в файл /etc/rc.conf следующие строки:

```
enable_quotas="YES"
check_quotas="NO"
```

Первая строка включает механизм квотирования, а вторая — отключает проверку квот при запуске, поскольку эта процедура занимает очень много времени.

Чтобы пользователи не потеряли свои данные, перезагрузитесь в однопользовательский режим (параметр ядра `single`). Вот теперь можно приступать к редакти-

рованию квот. Первым делом разрешим устанавливать квоты на разделе, который содержит файлы пользователей. Откройте файл /etc/fstab:

```
# ee /etc/fstab
```

Добавьте параметр `usrquota` к списку параметров каждого раздела, где нужно включить квоты:

```
/dev/ad0s1a    /      ufs          rw,userquota              1      1
/dev/ad0s1f    /usr   ufs          rw,userquota,groupquota   2      2
```

Параметр `usrquota` включает поддержку квот для отдельных пользователей. Если вам нужна поддержка квот групп пользователей, тогда добавьте параметр `groupquota`.

Далее лучше всего перезагрузить компьютер: `# reboot`

После перезагрузки можно приступить к установке квот. Если пользователей у вас много, то проще устанавливать квоты не для каждого пользователя отдельно, а создать несколько прототипов, которые будут использоваться для установки квот других пользователей. Определим лимиты для пользователя max:

```
# edquota -u max
Quotas for user max:
/: blocks in use: 65, limits (soft = 50, hard = 75)
inodes in use: 7, limits (soft = 50, hard = 60)
/usr: blocks in use: 0, limits (soft = 50, hard = 75)
inodes in use: 0, limits (soft = 50, hard = 60)
```

Разберемся, что есть что:

- `blocks` — место в блоках, используемое пользователем (1 блок = 1 Кбайт);
- `soft` — максимальное дисковое пространство (в блоках по 1 Кбайт), которое может занимать пользователь. Если вы включите период отсрочки (`grace period`), то пользователь получит только лишь сообщение о превышении квоты;
- `hard` — жесткое ограничение, эту квоту пользователь превысить не может, даже если включен период отсрочки. Предположим, что вы хотите отдать пользователю 500 Мбайт. В качестве жесткой квоты можно установить значение 500 Мбайт (или 500000 блоков), а в качестве мягкой — 495 Мбайт (495000 блоков). Когда пользователь превысит 495 Мбайт, он получит сообщение о превышении квоты, а вот когда будет превышена жесткая квота, то пользователь больше не сможет сохранять файлы.
- `inodes` — число используемых пользователем файлов (инодов).

> ***Примечание***
>
> Настоятельно рекомендую перед установкой квот выбрать (с помощью переменной окружения `EDITOR`) дружественный текстовый редактор, иначе будете использовать неудобный редактор vi.

Чтобы установить период отсрочки, используется команда `edquota -t`. При указании периода отсрочки вы можете использовать названия единиц изменения время на английском:

- seconds — секунды;
- minutes — минуты;
- hours — часы;
- days — дни;
- weeks — недели;
- months — месяцы.

Например:

- 24hours — 24 часа;
- 2days — 2 дня;
- 1weeks — 1 неделя.

Значительно упрощают задание квот так называемые *прототипы*. Например, вы задали ограничение для пользователя den. Но у вас есть еще несколько пользователей, для которых нужно задать такие же ограничения. Вы можете использовать квоту пользователя den в качестве прототипа:

```
# edquota -p den user1
# edquota -p den user2
...
```

Вместо имен пользователей можно использовать UID пользователя и даже диапазон UID, например:

```
# edquota -p den 1001-9999
```



Глава 16

Загрузка и инициализация системы

16.1. Процесс загрузки FreeBSD

Процесс загрузки BSD похож на процесс загрузки любой другой операционной системы. Вы включаете питание компьютера, BIOS запускает процедуру самотестирования компьютера — POST (Power On Self Test). Затем BIOS пытается запустить операционную систему — ей все равно, какая операционная система у вас установлена, она должна загрузить код загрузчика операционной системы, находящийся в главной загрузочной записи (Master Boot Record, MBR), а уж загрузчик пусть сам разбирается, что и как ему загружать. Благодаря такой схеме допускается использование нескольких операционных систем на одном компьютере. BIOS запускает загрузчик операционной системы, а он выводит меню выбора операционной системы (рис. 16.1).

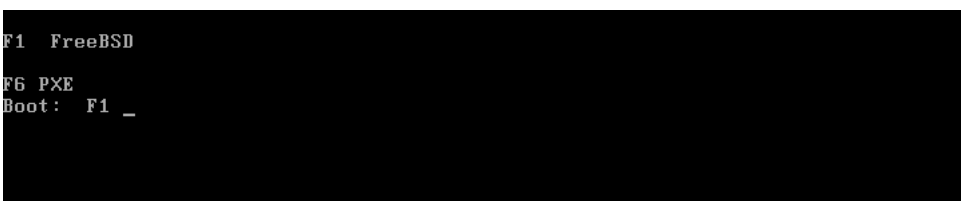


Рис. 16.1. Меню выбора операционной системы загрузчика BootMgr

Для загрузки FreeBSD нужно нажать клавишу <F1>, для запуска PXE (Preboot Execution Environment — среда сетевой загрузки компьютеров с помощью сетевой карты, но без использования жесткого диска) — клавишу <F6>. Сетевая загрузка нас сейчас мало беспокоит, поэтому нужно выбрать первый вариант — нажать клавишу <F1>.

Вы увидите меню выбора вариантов загрузчика FreeBSD (рис. 16.2):

- **Boot FreeBSD** — самая обычная загрузка системы, вариант по умолчанию;
- **Boot FreeBSD with ACPI disabled** — загрузка FreeBSD без использования ACPI — используется на конфигурациях, имеющих проблемы с ACPI. Типичный пример проблемной конфигурации — компьютер с процессором AMD 64 и видеокартой от NVIDIA. Впрочем, такой вариант для сервера — редкость, это скорее игровая/домашняя машина, чем сервер сети;



Рис. 16.2. Меню загрузки FreeBSD

- **Boot FreeBSD is Safe Mode** — безопасный режим загрузки;
- **Boot FreeBSD in single user mode** — однопользовательский режим, удобно использовать его для восстановления системы;
- **Boot FreeBSD with verbose logging** — загрузка FreeBSD с расширенным протоколированием — воспользуйтесь, если стандартное протоколирование не позволяет выяснить причину сбоя;
- **Escape to loader prompt** — вызывает командную строку загрузчика (основные команды загрузчика представлены в табл. 16.1);
- **Reboot** — перезагрузка компьютера.

Таблица 16.1. *Основные команды загрузчика*

Команда	Описание
`boot [параметры] [ядро]`	Загружает указанный файл ядра (полезно использовать для загрузки старого ядра, если после перекомпиляции новое ядро не запускается)
`load [имя файла]`	Загружает ядро или модуль ядра, все аргументы будут переданы загружаемому файлу
`lsmod`	Выводит список загруженных модулей
`reboot`	Перезагружает компьютер
`unload`	Удаляет из памяти все загруженные модули
`help`	Выводит справку из файла /boot/loader.help

Существуют четыре стадии загрузки:

- первая стадия (boot0) — на этой стадии загрузчик BIOS запускает первый загрузочный блок boot0 из MBR, и вы видите меню выбора операционных систем.

Если вы не нажмете ни одну из функциональных клавиш, будет выбран вариант по умолчанию (F1);

- задача второй стадии (boot1) — найти и запустить третью стадию (boot2);
- на третьей стадии вы видите меню, изображенное на рис. 16.2. Параметры загрузчика, определяющие, как будет выглядеть это меню, находятся в файле boot/loader.conf. По умолчанию он пуст. Пример этого файла можно найти в файле /boot/defaults/loader.conf. Формат файла loader.conf похож на формат файла rc.conf — чуть дальше мы его рассмотрим. Каждая опция в файле /boot/defaults/loader.conf тщательно откомментирована, поэтому вы без проблем разберетесь с этим файлом самостоятельно. В этом же файле указываются команды загрузки модулей ядра, например, для загрузки модуля поддержки ZFS нужно добавить строку:

```
zfs_load="YES"
```

При желании загрузчик FreeBSD можно настроить так, как в Linux, — с красивой графической заставкой. Для включения графической заставки установите следующие опции так:

```
splash_bmp_load="YES"
splash_pcx_load="NO"
vesa_load="YES"
bitmap_load="YES"
bitmap_name="splash.bmp"  # Путь к BMP-файлу
```

- четвертая стадия — это загрузка ядра операционной системы и передача ядру параметров. Например, параметр `-s` означает однопользовательский режим, а параметр `-v` — более подробное протоколирование при загрузке, что равносильно выбору опции меню **Boot FreeBSD with verbose logging** (см. рис. 16.2).

При загрузке ядра выводятся диагностические сообщения — ядро сообщает нам о ходе загрузке и о найденных устройствах. Эти сообщения пролетают по экрану так быстро, что вы не успеваете их прочитать. Но не беспокойтесь — после загрузки системы вы можете изучить их в комфортной обстановке и без всякой спешки:

```
# dmesg | less
```

Приводить здесь вывод команды `dmesg` не буду — все равно он привязан к моему "железу". Лучше рассмотрим несколько полезных команд, позволяющих получить из вывода `dmesg` полезную информацию (табл. 16.2).

Таблица 16.2. *Примеры получения информации с помощью* `dmesg`

Команда	Описание
`# dmesg | grep Ethernet`	Получить имена сетевых адаптеров и их MAC-адреса
`# dmesg | grep memory`	Получить информацию об оперативной памяти: real memory — всего установлено памяти, avail memory — доступно памяти
`# dmesg | grep Hard`	Обычно получает информацию о жестком диске

Таблица 16.2 *(окончание)*

Команда	Описание
`# dmesg | grep ad0` или `dmesg | grep da0`	Получает информацию о первом жестком диске
`# dmesg | grep usb`	Информация об USB-устройствах и контроллерах
`# dmesg | grep CPU:`	Информация об используемом процессоре
`# dmesg | grep "root from"`	Имя раздела, содержащего корневую файловую систему и ее тип

При загрузке в однопользовательском режиме не запрашивается пароль root, что не очень хорошо с точки зрения безопасности. А все потому, что консоль считается безопасной. Откройте файл /etc/ttys, найдите в нем строку:

```
console none  unknown off secure
```

Данная консоль используется при загрузке в однопользовательском режиме. Измените ее тип на `insecure`:

```
console none  unknown off insecure
```

После загрузки ядро запускает систему инициализации (программу /sbin/init), которая выполняет всю дальнейшую инициализацию системы: загружает сетевые службы, устанавливает глобальные параметры системы (обработка файла rc.conf), производит монтирование файловых систем, указанных в файле /etc/fstab.

16.2. Сценарии инициализации

Откуда система знает, какие службы нужно запускать? Информация о запускаемых службах, а также некоторые системные параметры (шрифт консоли, имя узла, параметры сетевых интерфейсов и т. д.) хранятся в файле /etc/rc.conf. По умолчанию этот файл пуст или содержит данные сетевых интерфейсов, если вы их настроили при установке системы.

Пример файла rc.conf находится в каталоге /etc/defaults. В файле /etc/defaults/rc.conf описаны опции, которые могут содержаться в файле rc.conf, и я настоятельно рекомендую вам изучить этот файл. А вот редактировать файл /etc/defaults/rc.conf не нужно — это только пример. Однако вы можете скопировать необходимые вам опции из файла /etc/defaults/rc.conf в файл /etc/rc.conf.

> ***Внимание!***
>
> Копировать все содержимое файла /etc/defaults/rc.conf в файл /etc/rc.conf ни в коем случае нельзя, поскольку это нарушит нормальную загрузку системы. При бездумном копировании в файл rc.conf могут попасть опции вызова служб, которые вообще не установлены в вашей системе, в результате при загрузке системы возникнет ошибка.

Приводить здесь файл /etc/defaults/rc.conf нет смысла — каждая опция тщательно откомментирована, и это бы существенно увеличило размер книги — настолько этот файл велик. Но некоторые полезные опции файла rc.conf я все-таки приведу:

```
# На все вопросы, задаваемые утилитой fsck, будет автоматически
# дан ответ 'Y'
```

```
fsck_y_enable="yes"
# Запрет запуска fsck в фоновом режиме (чтобы вы видели
# все изменения, вносимые утилитой проверки диска)
background_fsck="NO"
# Запуск демона мыши в консоли
moused_enable="YES"
# Имя файла подкачки
swapfile="/usr/swap0"
```

Последняя строка загружает файл подкачки /swap. Дело в том, если основного раздела подкачки оказалось мало, вам предоставляется возможность создать дополнительный файл подкачки. Только убедитесь, что ядро системы *(см. главу 21)* собрано с опцией:

```
device    md    # Memory "disks"
```

Если это не так, придется ядро перекомпилировать. Затем создайте файл размером 512 Мбайт:

```
# dd if=/dev/zero of=/usr/swap0 bs=1024k count=512
```

Установите права доступа:

```
# chmod 0600 /usr/swap0
```

Вот теперь и нужно добавить ту самую последнюю строку в файл rc.conf, после чего перезагрузить компьютер. Можно обойтись и без перезагрузки:

```
# mdconfig -a -t vnode -f /usr/swap0 -u 0 && swapon /dev/md0
```

16.3. Планировщики заданий

Очень часто приходится периодически выполнять одни и те же действия. Например, ежедневно обновлять антивирусные базы или каждые 30 минут проверять почту. Можно выполнять эти действия самому, но это вариант не лучший. Представьте, что ваш рабочий день будет начинаться с команды запуска программы обновления антивируса, а каждые 30 минут вам придется запускать программу проверки почты. Во-первых, это не очень удобно, а во-вторых, можно легко забыть выполнить ту или иную команду. Например, в пятницу вечером вы можете забыть выполнить команду создания резервной копии, а в понедельник утром что-то случится с сервером, и вы не досчитаетесь всего пользовательского каталога. Не очень приятно, правда?

Во всех UNIX-системах (и FreeBSD не исключение) имеется специальный демон crond, позволяющий выполнять программы по расписанию. Для запуска демона crond в файл /etc/rc.conf нужно добавить строку:

```
cron_enable="YES"
```

Откройте конфигурационный файл демона crond — /etc/crontab (листинг 16.1).

Листинг 16.1. Пример файла /etc/crontab

```
# $FreeBSD: src/etc/crontab,v 1.32 2002/11/22 16:13:39 tom Exp $
#
SHELL=/bin/sh
PATH=/etc:/bin:/sbin:/usr/bin:/usr/sbin
HOME=/var/log
#
#
#minute hour    mday    month   wday    who command
#
#
*/5 *   *   *   *   root    /usr/libexec/atrun
```

Параметр `SHELL` задает имя программы-оболочки, параметр `PATH` — путь поиска программ, `MAILTO` — имя пользователя, которому будет отправлен отчет о выполнении расписания (по умолчанию параметр `MAILTO` не указывается), а `HOME` — домашний каталог crond.

Но главное — не эти параметры, а сама таблица расписаний, занимающая в нашем случае всего одну строку — последнюю. Согласно этой таблице, каждые 5 минут будет выполняться программа /usr/libexec/atrun.

Формат записей таблицы расписаний следующий:

```
минуты (0–59) часы (0–23) день (1–31) месяц (1–12) день_недели (0–6, 0 — Вс)
пользователь команда
```

Пусть мы хотим, чтобы одна программа выполнялась в 7:00, а другая в 7:20. Чтобы реализовать наше расписание, следует добавить в файл /etc/crontab следующие строки:

```
0       7       *       *       *       root /usr/bin/command1 arguments
20      7       *       *       *       root /usr/bin/command2 arguments
```

Файл /etc/crontab имеет право редактировать только пользователь root. После редактирования не нужно предпринимать никаких действий — crond сразу же узнает, что вы изменили расписание.

Файл /etc/crontab содержит общесистемную таблицу расписаний. Однако с помощью команды `crontab` можно изменить таблицу расписаний для определенного пользователя:

```
# crontab -u имя_пользователя
```

Создадим файл /var/cron/allow:

```
# touch /var/cron/allow && chmod 600 /var/cron/allow
```

В этом файле нужно перечислить пользователей (по одному в каждой строке), которым разрешено использовать команду `crontab`. Не забудьте в этот список включить пользователя root!

Пользовательские расписания хранятся в каталоге /var/cron/tabs. Формат пользовательских файлов такой же, как и файла /etc/crontab, за исключением отсутствия поля `who` (содержит имя пользователя), то есть формат будет следующим:

```
минуты часы день месяц день_недели команда
```

16.4. Настройка синхронизации времени

Для синхронизации времени с удаленной машиной при загрузке системы добавьте в файл /etc/rc.conf следующие строки:

```
ntpdate_enable="YES"
ntpdate_flags="-b IP-адрес-сервера-времени"
```

Если есть необходимость в периодической синхронизации времени, тогда добавьте следующее задание в таблицу crontab:

```
*/55 *  *  *  *  /usr/sbin/ntpdate -B IP-сервера > /dev/null 2>&1
```

16.5. Тюнинг системы с помощью файла sysctl.conf

В файле /etc/sysctl.conf содержатся переменные, способные влиять на различные параметры системы: от параметров ядра до параметров сети. Например, следующая строка в файле sysctl.conf устанавливает время жизни (TTL) пакета, равное 128, что позволяет "замаскироваться" под Windows:

```
net.inet.ip.ttl=128
```

Список полезных sysctl-переменных можно найти по адресам:

- **http://www.opennet.ru/tips/info/847.shtml**;
- **http://sysctl.enderunix.org/**.

Глава 17



Процессы

17.1. Аварийное завершение процесса

Каждому процессу в FreeBSD присваивается уникальный номер — идентификатор процесса (PID, Process ID). Зная ID процесса, вы можете управлять процессом, а именно — завершить процесс или изменить приоритет процесса. Принудительное завершение процесса необходимо, если процесс завис, и его нельзя завершить обычным образом. А изменение приоритета может понадобиться, если вы хотите, чтобы процесс доделал свою работу быстрее.

Предположим, у вас зависла какая-то программа, например, пусть это будет файловый менеджер mc. Хоть это и маловероятно (не помню, чтобы он когда-нибудь зависал), но для примера пусть будет так. Принудительно завершить ("убить") процесс можно с помощью команды `kill`. Формат ее вызова следующий:

```
kill [параметры] PID
```

`PID` (Process ID) — это идентификатор процесса, который присваивается процессу системой и уникален для каждого процесса. Но мы знаем только имя процесса (имя команды), но не знаем идентификатор процесса. Узнать идентификатор процесса позволяет команда `ps`. Предположим, что mc находится на первой консоли. Поскольку он завис, вы не можете более использовать консоль, и вам нужно переключиться на вторую консоль (клавиатурной комбинацией <Alt>+<F2>). Зарегистрировавшись на второй консоли, введите команду `ps`. Вы увидите список процессов, запущенных от имени вашего пользователя. Поскольку команду `ps` я ввел от имени root, то кроме оболочки csh (на консоли V0 и V1), самой ps и mc, будут отображены и копии программы getty (ожидает входа пользователя) на консолях V2–V7 (рис. 17.1).

Команда `ps` выводит идентификатор процесса (PID), терминал, к которому привязан процесс (TT), состояние процесса, процессорное время (а не время выполнения) и команду запуска процесса.

У команды `ps` много параметров, и с ними вы можете ознакомиться на странице руководства (`man ps`), а в табл. 17.1 собраны самые полезные параметры этой команды.

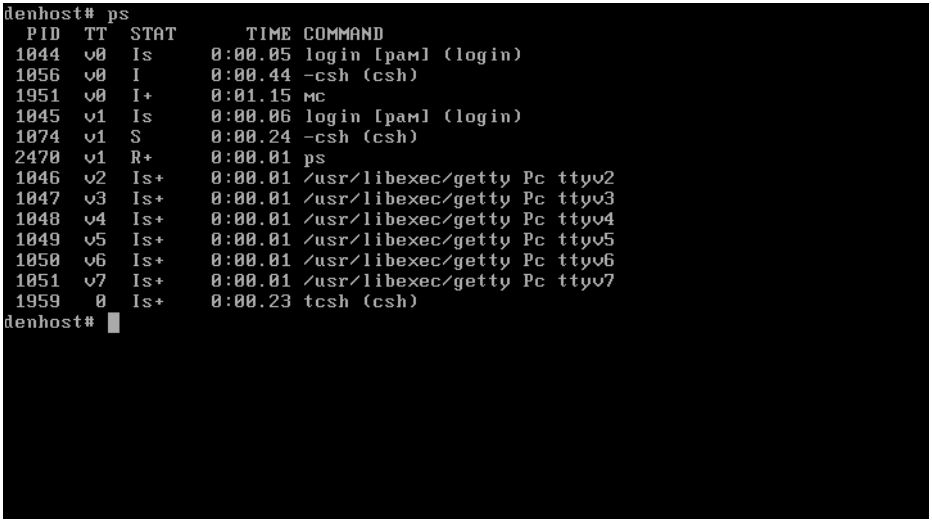


Рис. 17.1. Результат выполнения команды `ps` от имени root

Таблица 17.1. *Самые полезные параметры команды* `ps`

Параметр	Описание
`-a`	Отображает информацию о процессах всех пользователей, а не только о собственных
`-c`	Вместо полной команды в последнем столбце будет отображено только имя исполнимого файла
`-C`	Изменяет способ вычисления процессорного времени (на результат практически не влияет)
`-e`	Отображает окружение
`-G gid`	Отображает информацию о процессах, принадлежащих пользователям из группы с идентификатором gid (gid — это не имя группы, а число, идентификатор)
`-h`	Если запущено слишком много процессов, и все они не помещаются на одном экране, повторяет заголовок через каждую "страницу"
`-r`	Сортировка по использованию процессора, а не по идентификатору и номеру терминала
`-U пользователь`	Отображает процессы, принадлежащие определенному пользователю, например, ps -U den
`-w`	"Широкий" вывод, используются 132 колонки
`-X`	Показывает только процессы, привязанные к терминалу (по умолчанию)
`-x`	Показывает процессы, не привязанные к терминалу (например, сетевые демоны)

Таблица 17.1 *(окончание)*

Параметр	Описание
`-l`	Показывает дополнительную информацию о процессе: идентификатор родительского процесса (PPID), приоритет, использование памяти и пр.
`-u`	Еще более полезный параметр: позволяет узнать, какой пользователь владеет процессом, просмотреть информацию об использовании оперативной памяти процессом и т. д. Полезен при следующем наборе параметров: -uax

Теперь, когда мы знаем PID нашего процесса, мы можем его "убить", например:

```
# kill 1951
```

Перейдите на первую консоль после выполнения этой команды — mc на ней уже не будет. Если выполнить команду `ps -a`, то в списке процессов mc тоже не будет.

Проще всего вычислить PID процесса с помощью следующей команды:

```
# ps -ax | grep <имя>
```

Например, `# ps -ax | grep firefox`.

Вообще-то все эти действия, связанные с вычислением PID процесса, мы рассмотрели только для того, чтобы познакомиться с командой `ps`.

Так что, если вы знаете только имя процесса, гораздо удобнее использовать команду:

```
# killall <имя процесса>
```

Но имейте в виду, что данная команда завершит все экземпляры данного процесса. А вполне может быть, что у нас на одной консоли находится mc, который нужно "убить", а на другой — нормально работающий mc. Команда `killall` "убьет" оба процесса.

Параметр `-u` команды `killall` позволяет "убить" процессы, принадлежащие заданному пользователю, например:

```
# killall -u denis
```

При выполнении команд `kill` и `killall` нужно помнить, что если вы работаете от имени обычного пользователя, они могут завершить только те процессы, которые принадлежат вам. А если вы работаете от имени пользователя root, то можете завершить любой процесс в системе.

С помощью команды `kill` можно не только "убить" процесс, но и послать процессу определенный сигнал — служебное сообщение о непредвиденной ситуации. Делается это так:

```
# kill <номер сигнала> PID
# kill -имя_сигнала PID
```

Вы можете указать как имя, так и номер сигнала — на ваше усмотрение. Таблица 17.2 содержит информацию о сигналах в FreeBSD.

Таблица 17.2. *Сигналы процессов FreeBSD*

№	Название	Описание
01	HUP	Освободить линию (hangup)
02	INT	Прерывание (interrupt)
03	QUIT	Выход (quit)
04	ILL	Некорректная команда (illegal instruction)
05	TRAP	Прерывание трассировки (trace trap)
06	IOT или ABRT	Аварийное завершение процесса
07	EMT	Машинная команда EMT
08	FPE	Исключительная ситуация при выполнении операции с вещественными числами (floating-point exception)
09	KILL	Уничтожение процесса. Используется по умолчанию, если номер сигнала не задан
10	BUS	Ошибка шины
11	SEGV	Некорректное обращение к сегменту памяти (segmentation violation)
12	SYS	Некорректный параметр системного вызова
13	PIPE	Запись в канал, из которого некому читать
14	ALRM	Сигнал тревоги
15	TERM	Программный сигнал завершения. Самое обычное завершение работы
16	USR1	Пользовательский сигнал 1
17	USR2	Пользовательский сигнал 2
18	CLD	Завершить порожденный процесс
19	PWR	Ошибка питания

Обычно можно не указывать номер сигнала, по умолчанию будет использоваться сигнал KILL.

17.2. Программа top: кто больше всех расходует процессорное время?

Иногда бывает, что система ужасно "тормозит" — весь день работала нормально, а вдруг начала "притормаживать".

Если вы даже не догадываетесь, из-за чего это случилось, нужно использовать программу top (рис. 17.2) — она выводит список процессов с сортировкой по процессорному времени. То есть на вершине списка будет процесс, который занимает больше процессорного времени, чем сама система. Вероятно, из-за него и происходит эффект "торможения".

```
last pid:  7107;  load averages:  1.03,  0.54,  0.22    up 0+00:33:39  17:38:41
28 processes:  2 running, 26 sleeping
CPU: 85.8% user,  0.0% nice, 13.5% system,  0.7% interrupt,  0.0% idle
Мем: 45M Active, 22M Inact, 40M Wired, 148K Cache, 34M Buf, 131M Free
Swap: 479M Total, 479M Free

  PID USERNAME  THR PRI NICE   SIZE    RES STATE    TIME   WCPU COMMAND
 7106 root        1 109    0 42340K 40132K RUN      0:10 60.99% cc1
 1121 root        1  44    0  3684K  1912K RUN      0:01  0.00% top
 7107 root        1  46    0  2912K  1712K piperd   0:00  0.00% as
 1052 root        1  44    0  5652K  2380K pause    0:00  0.00% csh
 1187 root        1  76    0  2912K  1108K wait     0:00  0.00% make
 1046 root        1  44    0  3812K  1776K wait     0:00  0.00% login
 1138 root        1  44    0  5652K  2604K pause    0:00  0.00% csh
 1148 root        1  76    0  2912K  1036K wait     0:00  0.00% make
  969 root        1  44    0  6080K  3228K select   0:00  0.00% sendmail
  679 root        1  44    0  3348K  1340K select   0:00  0.00% syslogd
 1044 root        1  44    0  3812K  1736K wait     0:00  0.00% login
  980 root        1  44    0  3376K  1372K nanslp   0:00  0.00% cron
 1045 root        1  44    0  3812K  1772K wait     0:00  0.00% login
 7032 root        1  76    0  1888K   624K wait     0:00  0.00% make
  509 root        1  53    0  3288K  1288K select   0:00  0.00% dhclient
 1049 root        1  76    0  3348K  1156K ttyin    0:00  0.00% getty
 6778 root        1  76    0  1888K   608K wait     0:00  0.00% make
 1051 root        1  76    0  3348K  1156K ttyin    0:00  0.00% getty
```

Рис. 17.2. Программа top

На рис. 17.2 показано, что больше всего процессорного времени (60,99%) занимает программа cc1 (я запустил сборку lynx из портов). Выйти из программы top можно, нажав клавишу <Q>. Кроме команды <Q> действуют следующие клавиши:

- <U> — показывает только пользовательские процессы (появится небольшое текстовое поле, и вы сможете ввести имя интересующего вас пользователя, для отображения процессов всех пользователей введите плюс: +);
- <H> — получить справку по остальным командам программы top.

Назначение столбцов программы top указано в табл. 17.3.

Таблица 17.3. *Назначение столбцов программы top*

Столбец	Описание
PID	Идентификатор процесса
USERNAME	Имя пользователя, запустившего процесс
THR	Приложения бывают многопоточными. Если вы до этого работали только в Windows, то лучший пример многопоточного (multithread) приложения — это любой менеджер загрузки файла. Файл условно разбивается на части, а приложение запускает несколько потоков — каждый поток "качает" свою часть файла. В результате существенно сокращается время закачки. Так вот, если в столбце THR значение, превышающее 1, данный процесс является многопоточным
PR	Приоритет процесса
NICE	Показатель nice (*см. разд. 17.3*), может принимать значения от -20 до 20
SIZE	Общий выделенный размер процесса, в том числе размеры сегмента кода, данных и стека. Значения SIZE и RES выводятся в килобайтах

Таблица 17.3 *(окончание)*

Столбец	Описание
RES	Размер процесса, не перемещенный в область подкачки (в Кбайт). Этот размер равен размерам сегментов кода и данных, то есть RES = CODE + DATA. Значения SIZE и RES отличаются. Если вы привыкли к диспетчеру задач Windows, тогда ориентируйтесь на столбец RES — это то же, что и столбец **Память** в диспетчере задач Windows
STATE	Состояние процесса: • **START** — процесс запускается; • **RUN** — процесс выполняется; • **WAIT** — состояние ожидания; • **ZOMB** — зомби. Это когда процесс, по сути, завершен, но информация о нем еще не удалена из таблицы процессов по какой-то причине — обычно по причине кривизны устройства /dev/hands программиста, написавшего программу ☺; • **STOP** — процесс остановлен; • **PAUSE** — процесс приостановлен; • **SLEEP** — процесс "спит". О других состояниях можно прочитать в `man top`
TIME	Процессорное время, израсходованное с момента запуска процесса
WCPU	Загрузка процессора в данный момент (в процентах). Именно по этому столбцу выполняется сортировка списка процессов
COMMAND	Команда, которая использовалась для запуска процесса (обычно имя исполнимого файла процесса)

Первые пять строчек вывода top тоже очень интересны. Они содержат полезную информацию о работе всей системы, а не отдельно взятого процесса. Так, в первой строчке выводится идентификатор последнего запущенного процесса **(last pid)**, общая нагрузка системы (**load average**) — за одну минуту, пять минут и пятнадцать минут соответственно (три числа после **load avevrage**), время, которое работает система с момента загрузки (**up**), текущее время.

Во второй строке выводится информация и процессах: общее количество процессов (на рис. 17.2 — **28 processes**), количество запущенных (**running**), "спящих" (**sleeping**), остановленных (**stopped**) процессов и процессов-зомби (**zombie**).

> ***Пояснение***
>
> Зомби — это уже "мертвый" (завершенный процесс), но информация о нем еще не удалена из таблицы процессов. Если вы обладаете минимальными навыками программирования на C, вам будет интересна следующая статья: **http://www.dkws.org.ua/index.php?page=show&file=a/dev/process2**. В ней я показываю, как можно создать зомби. Заодно поймете, как так получается, что система не успевает удалять информацию о процессе из служебной таблицы.

Третья строка выводит информацию о распределении процессорного времени, то есть сколько времени (в процентном соотношении) процессор пребывает в том

или ином состоянии: **user** (пользовательские задачи), **nice** (выполнение процессов с измененным показателем **nice**), **system** (системные задачи), **interrupt** (обработка прерываний) и **idle** (простой).

Последние две строки выводят информацию об использовании оперативной памяти и подкачки (виртуальной памяти). Обычно нас интересует значение **Free**, содержащее количество мегабайтов свободной памяти/подкачки.

17.3. Изменение приоритета процесса

Предположим, вы работаете с видео, и вам нужно перекодировать файл из одного видеоформата в другой. Конвертирование видео занимает много процессорного времени, а хотелось бы все сделать как можно быстрее и уйти раньше домой. Тогда вам поможет программа nice — она позволяет запустить любую программу с указанным приоритетом. Ясно — чем выше приоритет, тем быстрее будет выполняться программа. Формат вызова команды следующий:

```
nice -n <приоритет> команда аргументы
```

Максимальный приоритет задается числом `-20`, а минимальный — числом `19`. Приоритет по умолчанию равен 10.

Если процесс уже запущен, тогда для изменения его приоритета можно использовать команду `renice`:

```
renice -n <приоритет> -p PID
```

17.4. Фоновое выполнение процессов

Команда `bg` может перевести процесс в фоновый режим, `fg` — переводит процесс обратно — "на передний план", а команда `jobs` отображает список задач, выполняющихся в фоновом режиме. Подробно о командах `bg`, `fg` и `jobs` вы можете прочитать в справочной системе.



Глава 18

Установка программного обеспечения: порты и пакеты

18.1. Введение в пакеты и порты

При разработке операционной системы перед ее создателями встает проблема распространения программного обеспечения. Понятно, что разработчики операционной системы физически не могут создать для нее все необходимые программы. В мире нет такой операционной системы, где был бы набор программ, способный удовлетворить потребности всех пользователей системы. Поэтому нужно придумать способ установки в созданную систему программ сторонних разработчиков.

Самый простой способ — это распространение исходного кода программы. Так раньше они и распространялись. Вы получали не откомпилированную программу, а ее исходный код. При наличии компилятора, некоторых вспомогательных утилит и "прямых рук" программу можно было откомпилировать. Теоретически такой способ установки возможен в любой операционной системе — будь то FreeBSD или Windows. В некоторых системах, например, в IRIX, этот способ используется до сих пор как единственный возможный. Правда, самой IRIX уже нет — разработка IRIX была прекращена компанией SGI в 2006 году, но поддержка пользователей планируется до 2013 года.

Установка программ путем компиляции исходного кода — не совсем хороший способ. И дело даже не в том, что на каждую машину придется установить компилятор и вспомогательные программы. А дело в наличии "прямых рук". Не всегда программа собирается без ошибок. Возможно, придется модифицировать ее исходный код, чтобы она установилась именно на вашей системе. А это требует наличия навыков программиста. Но ведь пользователь на то он и пользователь, что не знает языки программирования, — он по определению не программист. Следовательно, такой способ не является универсальным и подходит далеко не всем. К тому же у компиляции исходного кода есть еще один весомый недостаток — компиляция требует времени, порой очень существенного.

Пришлось искать другой способ. Можно откомпилировать исходный текст программы и распространять уже ее в откомпилированном виде. Довольно хорошее решение. Пользователю нужно скачать "бинарник", подходящий для его архитектуры, и скопировать в определенное место файловой системы. Но что делать, если программа состоит не из одного "бинарника", а из нескольких плюс вспомогательные файлы (документация, файлы конфигурации и т. д.)? Тогда стали распространять

программы в архивах. Очень удобно. До сих пор даже для Windows некоторые программы распространяются в виде архива — вы можете распаковать архив туда, куда вам будет удобно, и запустить программу.

Тем не менее, в Windows пошли иным путем. В дистрибутив каждой распространяемой программы включается программа установки, которая выполняет все необходимые действия: извлекает из CAB-файлов (архивов) или из самого исполнимого файла программы (который является самораспаковывающимся архивом) все необходимые файлы во временный каталог, затем копирует их в каталог, указанный пользователем, после чего вносит необходимые изменения в реестр системы и т. д.

Разработчики UNIX-подобных систем оказались как всегда впереди планеты всей — они и не думали создавать программу установки. А зачем? Каждый раз копировать, по сути, одинаковый код, рисующий полоску процесса, вносящий изменения в реестр и пр. Ведь это пустая трата дискового пространства. И поступили гениально просто. В обычный архив они добавили несколько служебных файлов, описывающих действия, которые нужны выполнить менеджеру пакетов для установки программы: какие файлы и куда скопировать, какие пакеты доустановить и т. д. Получилось, что менеджер пакетов для всех устанавливаемых программ всего один, а не распространяется с каждым дистрибутивом программы-приложения.

В FreeBSD вы можете как установить программу из исходных кодов, так и установить пакет с уже откомпилированной программой. Но в каждом из случаев возможны дополнительные варианты. Сначала разберемся с исходным кодом. Вы можете скачать исходный код программы самостоятельно, распаковать его и откомпилировать. А можете воспользоваться портами. *Порты* — это коллекции исходного кода программ. Все, что нужно сделать для установки той или иной программы — это перейти в нужный подкаталог каталога /usr/ports и ввести команду `make install clean`.

Теперь о пакетах. Источником пакетов может выступать дистрибутивный DVD, FTP-сервер и т. п. Достаточно скачать пакет или указать путь к нему (как локальный, так и удаленный, если пакет находится на FTP-сервере) и дать команду его установить. Как видите, все не так и сложно, поэтому самое время перейти от теории к практике.

18.2. Установка из портов

По сути, установка из портов — это компиляция исходного кода программ. Для установки программы из портов необходимо наличие активного интернет-соединения, поскольку исходный код программы загружается из Интернета, а затем на компьютере производится его компиляция.

Практически для каждого порта уже существует откомпилированный пакет, который можно установить командой `pkg_add`, что существенно сокращает время установки программы. Но в установке из портов есть и свои преимущества — каждый порт содержит все самые последние патчи, поэтому можно быть уверенным, что вы установите новейшую версию программы. Но за это придется расплатиться собственным временем — иногда таким, что терпения не хватает — так долго занимает установка из портов.

18.2.1. Установка порта

Для установки программы из портов нужно выполнить две команды. Первая — это перейти в каталог нужного вам порта. Как правило, этот каталог содержится в каталоге /usr/ports, например:

```
# cd /usr/ports/www/lynx
```

После этого следует ввести команду:

```
# make install clean
```

Программа `make` выполнит сборку целей `install` и `clean`. Как правило, цель `install` производит компиляцию и установку программы, а цель `clean` — удаление "мусора" (ненужных файлов), образовавшегося в процессе компиляции.

Напомню только, что перед установкой из портов нужно активировать интернет-соединение, поскольку исходные коды будут загружены из Интернета.

Для установки порта также можно воспользоваться командой `portinstall`:

```
# portinstall <имя порта>
```

У команды `portinstall` много параметров, поэтому не поленитесь и загляните в `man portinstall`.

18.2.2. Удаление и переустановка порта

Иногда программу по тем или иным причинам необходимо переустановить. Для этого нужно перейти в каталог порта и ввести команду:

```
# make deinstall reinstall clean
```

В этом случае сначала будет выполнена деинсталляция, а затем — переустановка и очистка.

Одна из причин переустановки — это неправильный набор опций. Дело в том, что при первой установке программы обычно вы видите меню, где нужно выбрать опции, с которыми программа будет откомпилирована. Может случиться так, что вы забудете включить ту или иную опцию, а когда это обнаружится, программа уже окажется установленной. В этом случае последовательность действий по переустановке программы будет иной. Сначала следует перейти в каталог порта, затем — деинсталлировать программу:

```
# make deinstall
```

А потом выполнить цели `config` (отображает то самое окошко с опциями), `install` и `clean`:

```
# make config install clean
```

18.2.3. Установка коллекции портов

Коллекция портов устанавливается, как правило, при установке системы. Если при установке системы вы по незнанию отказались от установки коллекции портов, установим порты вручную. Проще всего для этого использовать команду:

```
# sysinstall
```

В открывшемся окне конфигуратора выберите опцию **Configure** (рис. 18.1), затем — **Distributions** (рис. 18.2), после чего — **ports** (рис. 18.3). Нажмите **OK** — откроется окно выбора источника установки (рис. 18.4). Для установки с дистрибутивного диска нужно выбрать **CD/DVD**. Для получения самой последней версии коллекции портов — выберите **FTP** для загрузки портов из Интернета. После выбора носителя начнется установка коллекции портов. По завершении установки выберите опцию **Exit**, а затем нажмите клавишу <X> для выхода из sysinstall.

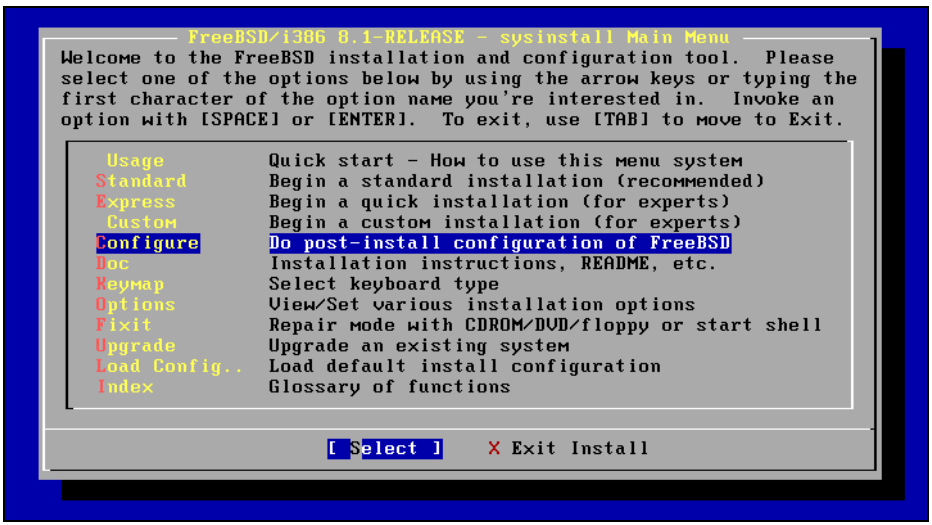


Рис. 18.1. Выберите **Configure**

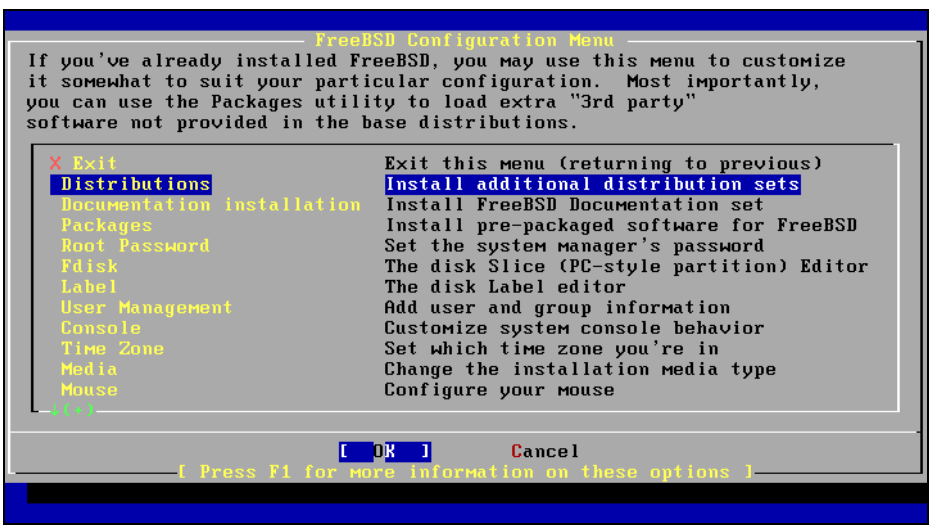


Рис. 18.2. Выберите **Distributions**

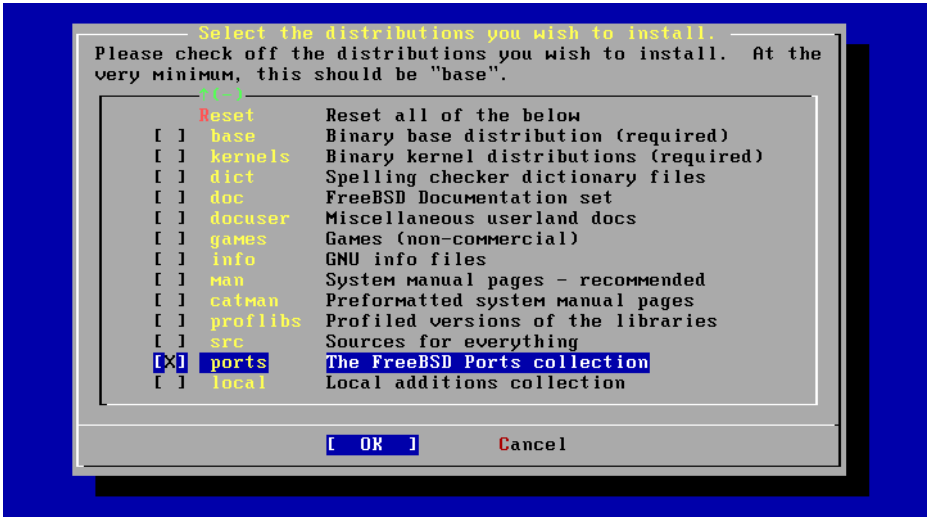


Рис. 18.3. Выберите **ports**

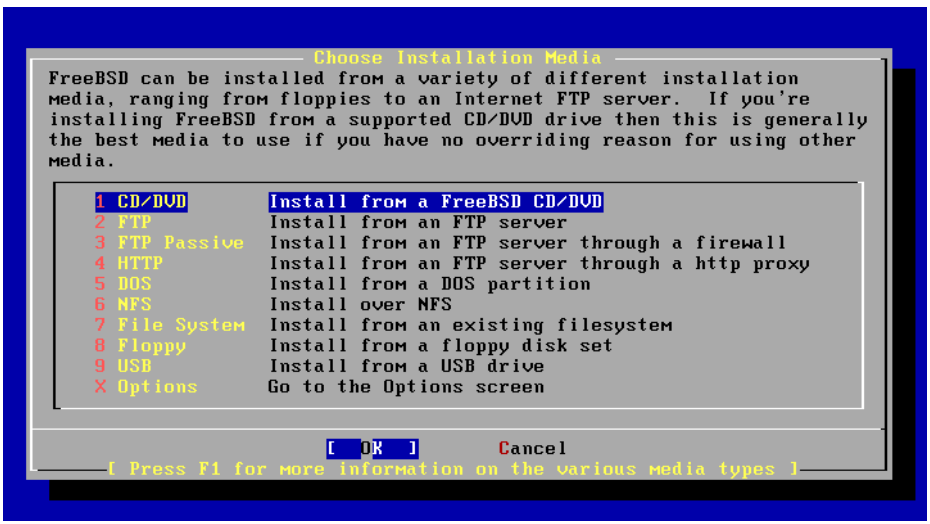


Рис. 18.4. Выберите носитель

18.2.4. Обновление коллекции портов

Установленную коллекцию портов нужно периодически — хотя бы раз в месяц — обновлять, иначе от нее не будет никакого толку. Для обновления портов используется программа cvsup. Чтобы не "собирать" ее из портов, введите команду:

```
# pkg_add -r cvsup
```

Так будет быстрее. Процесс установки программы cvsup изображен на рис. 18.5. Опция `-r` нужна для загрузки пакетов из интернет-хранилища. Для установки паке-

тов с дистрибутивного DVD эта опция не нужна, только не забудьте перед установкой пакета смонтировать DVD:

```
# mount /cdrom
# cd /cdrom/packages/All
# pkg_add cvsup
```



Рис. 18.5. Установка программы cvsup

Скопируйте файл /usr/share/examples/cvsup/ports-supfile (этот файл управляет обновлением портов) в свой домашний каталог или в каталог /usr/local/etc:

```
# cp /usr/share/examples/cvsup/ports-supfile /root
```

Для обновления коллекции портов выполните команду:

```
# /usr/local/bin/cvsup -L 2 /root/ports-supfile
```

Параметр `-L` можно вообще не указывать — он задает уровень протоколирования (2 — самый высокий уровень).

Примечание

Для получения пакета используется утилита fetch, но у вас могут возникнуть проблемы, если вы подключаетесь к Интернету не напрямую, а через прокси. В этом случае нужно установить переменные окружения `http_proxy` и `ftp_proxy`:

```
setenv  http_proxy      http://proxy.firma.ru:3128
setenv  ftp_proxy       http://proxy.firma.ru:3128
```

18.2.5. Описание каталога /usr/ports

В каталоге /usr/ports довольно много подкаталогов (рис. 18.6). В какой из них перейти, чтобы найти необходимый порт? В этом вам помогут данные табл. 18.1 с кратким описанием каждого из подкаталогов /usr/ports.

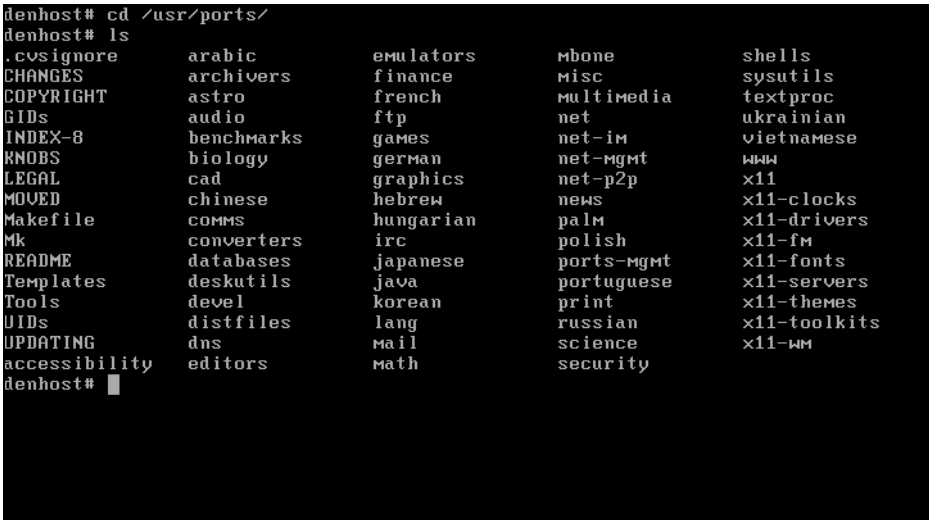


Рис. 18.6. Содержимое каталога /usr/ports

Таблица 18.1. *Описание подкаталогов /usr/ports*

Каталог	**Описание**
accessibility	Программы для пользователей с ограниченными возможностями
arabic	Программное обеспечение с поддержкой арабского языка
archivers	Архиваторы
astro	Приложения для астронома
audio	Программы для работы со звуком
benchmarks	Утилиты для измерения производительности системы
biology	Программы для биолога
cad	Системы автоматизированного проектирования
chinese	Программы с поддержкой китайского языка
comms	Коммуникационные утилиты
converters	Конвертеры: программы для преобразования форматов
databases	Программы для работы с базами данных
deskutils	Утилиты рабочего стола
devel	Для разработчика
dns	Клиентские и серверные утилиты DNS
editors	Самые обычные текстовые редакторы

Таблица 18.1 *(продолжение)*

Каталог	Описание
emulators	Эмуляторы: программы для эмуляции других операционных систем
finance	Приложения, связанные с финансами
french	Программы с поддержкой французского языка
ftp	FTP-серверы и клиенты
games	Игрушки
german	Программное обеспечение с поддержкой немецкого языка
gnome	Компоненты графической среды GNOME. В современной редакции портов этого каталога нет, но я его оставил в таблице специально — если вы работали с ранней версией FreeBSD и не можете найти привычный каталог. Теперь все, что касается поддержки GNOME, "переехало" в каталог x11
graphics	Программы для работы с графикой. Программу GIMP нужно искать в этом каталоге!
hebrew	Программное обеспечение с поддержкой иврита
hungarian	Программное обеспечение для венгерского рынка
ipv6	Программы с поддержкой IPv6
irc	Утилиты для Internet Relay Chat
japanese	Программы с поддержкой японского языка
java	Поддержка языка Java
kde	Компоненты графической среды K Desktop Environment. Этого каталога в современной редакции коллекции портов нет. Все, что касается KDE, вы найдете в каталоге x11
korean	Программы с поддержкой корейского языка
lang	Языки программирования
mail	Программы для работы с электронной почтой
math	Математическое программное обеспечение
mbone	Приложения и утилиты для MBONE
misc	Разное
multimedia	Программы для работы с мультимедиа
net	Программы для работы с сетью
net-im	Утилиты для мгновенного обмена сообщениями (BSD-версия "аськи" находится в этом каталоге)
net-mgmt	Утилиты управления сетью
net-p2p	Программы для работы в пиринговой сети (Torrent-клиент находится в этом каталоге)

Таблица 18.1 *(окончание)*

Каталог	Описание
news	Поддержка новостей USENET
palm	Программная поддержка линейки Palm(tm)
polish	ПО с поддержкой польского языка
ports-mgmt	Управление портами
portuguese	Программы, обеспечивающие поддержку португальского языка
print	Утилиты для работы с печатью
russian	Поддержка русского языка (например, средства проверки правописания находятся в этом каталоге)
science	Научное программное обеспечение
security	Программы, обеспечивающие безопасность системы
shells	Различные оболочки (tcsh, bash и т. д.)
spanish	ПО, обеспечивающее поддержку испанского языка
sysutils	Различные системные утилиты
textproc	Утилиты для обработки текста
ukrainian	Программы для поддержки украинского языка
vietnamese	Приложения, специфические для Вьетнама
www	Утилиты для WWW (браузеры, серверы HTTP и т. д.)
x11	Утилиты для X11. Здесь также следует искать GNOME и KDE
x11-clocks	Часы для X11
x11-drivers	Драйверы для X11
x11-fm	Файловые менеджеры для X11
x11-fonts	Шрифты для X11 и утилиты для работы с ними
x11-servers	Серверы для X1
x11-themes	Темы для X11
x11-toolkits	Инструменты разработки приложения для X11
x11-wm	Оконные менеджеры для X11. Здесь есть даже трехмерный рабочий стол Compiz
xfce	Порты, поддерживающие десктоп Xfce

Если вы не знаете, в каком именно каталоге находится нужный вам порт, можете перейти в каталог /usr/ports и ввести команду:

```
# make search name="имя порта"
```

Например:

```
# make search name="samba"
```

18.2.6. Обновление портов. Программа portupgrade

В *разд. 18.2.4* было показано, как обновить коллекцию портов. Сейчас разберемся, как обновить тот или иной порт. Ранее мы установили браузер lynx. Для его обновления можно было бы его деинсталлировать, а потом установить заново:

```
# cd /usr/ports/www/lynx
# make deinstall
# make config install clean
```

Но такой подход, сами понимаете, не очень соответствует современному представлении об обновлении программного обеспечения. Для начала определим, какие порты вообще нуждаются в обновлении. Введите команду:

```
# pkg_version -v | grep need
```

Вы получите список программ, которые следует обновить. Если программа нуждается в обновлении, напротив ее имени будет указано **need updating**, например:

epiphany-2.26.3_3 < needs updating (port has 2.26.3_4)
es-freebsd-doc-20090820 < needs updating (port has 20090913)
farsight2-0.0.14 < needs updating (port has 0.0.15)
fr-freebsd-doc-20090820 < needs updating (port has 20090913)
gdm-2.26.1_6 < needs updating (port has 2.26.1_7)

Из этого списка видно, что сейчас, например, установлена программа epiphany версии 2.26.3_3, но в портах есть уже версия 2.26.3_4, поэтому программа требует обновления.

Если у вас вывод указанной команды пуст, ни одна из программ не требует обновления. Можете в этом убедиться путем ввода команды `pkg_version -v`. Напротив имени каждой программы вы получите сообщение, что установленная версия программы соответствует версии из портов (рис. 18.7).

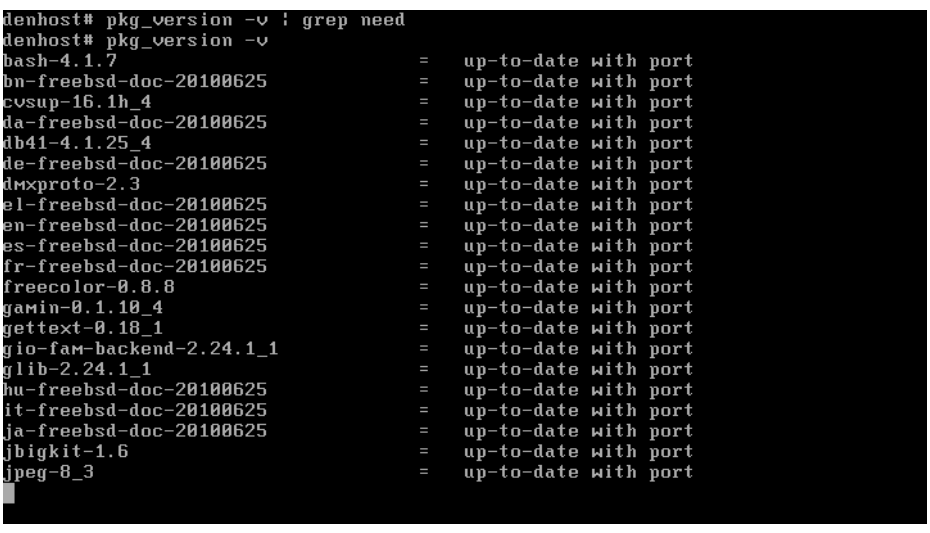


Рис. 18.7. Ни одна из программ не требует обновления



Рис. 18.8. Установка программы portupgrade

Для обновления порта используется программа portupgrade. Установите ее прямо сейчас, чтобы не тратить время на установку, когда понадобится обновить какой-нибудь порт. Быстрее будет, если установить эту программу из пакетов (рис. 18.8):

```
# pkg_add -r portupgrade
# rehash
```

После установки программы portupgrade нужно создать базу данных пакетов:

```
# pkgdb -F
```

Программа pkgdb выяснит перечень установленных программ и создаст базу данных в каталоге /var/db/pkg. Эту базу данных будет использовать программа portupgrade.

Чтобы обновить порт, достаточно указать его имя в качестве аргумента portupgrade, например:

```
# portupgrade epiphany
```

Чтобы portupgrade при обновлении использовала не порты, а пакеты (что существенно экономит время), задайте опцию `-P`:

```
# portupgrade -P epiphany
```

Если пакет с новой версией будет найден в каталогах, указанных в `PKG_PATH`, программа попытается загрузить их удаленно. А вот если на удаленном сервере тоже не окажется пакета с новой версией программы, тогда будут использоваться порты. Чтобы вообще запретить использование портов, включите опцию `-PP`:

```
# portupgrade -PP epiphany
```

Для обновления всех портов служит опция:

```
# portupgrade -a
```

Как видите, программу portupgrade удобно использовать не только для обновления портов, но и пакетов.

18.3. Установка программ из пакетов

Для установки программы из пакета (файла с "расширением" tbz) введите команду:

```
# pkg_add <имя пакета>
```

Команда `pkg_add` ищет пакет в локальных каталогах, указанных в переменной окружения `PKG_PATH`. Для установки пакета из удаленного источника (например, с FTP-сервера FreeBSD) укажите опцию `-r`, например:

```
# pkg_add -r mc
```

Если вы знаете точный путь к пакету (или URL пакета), можно указать его:

```
pkg_add ftp://ftp.freebsd.org/pub/FreeBSD/releases/i386/8.1-
RELEASE/packages/ftp/filezilla-3.3.3.tbz
```

Для удаления пакета используется команда `pkg_delete`:

```
# pkg_delete <имя_пакета>
```

Имя пакета нужно указывать вместе с версией:

```
# pkg_delete filezilla-3.3.3
```

Получить информацию о пакете можно с помощью команды `pkg_info`:

```
# pkg_info <имя пакета>
```

Помимо информации о том, какой файл и куда копируется, в пакетах содержится информация о зависимостях и конфликтах:

- **зависимости** — одна программа для своей работы может требовать какую-то библиотеку (без которой она не будет запускаться, поскольку использует функции этой библиотеки). Тогда в пакете указывается, что он *зависит* от другого пакета, содержащего библиотеку. При установке менеджер пакетов проверяет зависимости: если установлены не все пакеты, от которых зависит устанавливаемый пакет, установка будет прервана — пока вы не установите все необходимое. Правда, имеется возможность установки программы без удовлетворения зависимостей (тогда информация о зависимостях будет просто проигнорирована), но в большинстве случаев установленная таким образом программа работать не будет;
- **конфликты** — аналогично, одна программа может в системе конфликтовать с другой программой. Например, программы sendmail и postfix являются MTA-агентами (MTA, Mail Transfer Agent). Поскольку в системе может быть только один MTA-агент, установить можно или sendmail, или postfix, то есть пакет sendmail конфликтует с пакетом postfix и наоборот.

При просмотре информации о пакете информация о зависимостях и конфликтах указывается в следующих полях:

- **Required by** — поле содержит список пакетов, которым для нормальной работы требуется пакет, информацию о котором мы просматриваем;
- **Dependency** — поле содержит названия пакета, необходимого для работы этого пакета;
- **Conflicts** — содержит название несовместимого пакета.

```
denhost# pkg_info mc
pkg_info: can't find package 'mc' installed or in a file!
denhost# pkg_version -v | grep mc
libXdmcp-1.0.3                  =   up-to-date with port
mc-4.7.2_1                      =   up-to-date with port
denhost# pkg_info mc-4.7.2_1
Information for mc-4.7.2_1:

Comment:
Midnight Commander, a free Norton Commander Clone


Description:
GNU Midnight Commander is a user-friendly yet powerful file manager
and visual shell, useful to novice and guru alike. It provides a
clear, user-friendly, and somewhat protected interface to a Unix
system while making many frequent file operations more efficient and
preserving the full power of the command prompt. You will wonder how
you could ever live without it.

WWW: http://www.midnight-commander.org


denhost#
```

Рис. 18.9. Информация о пакете

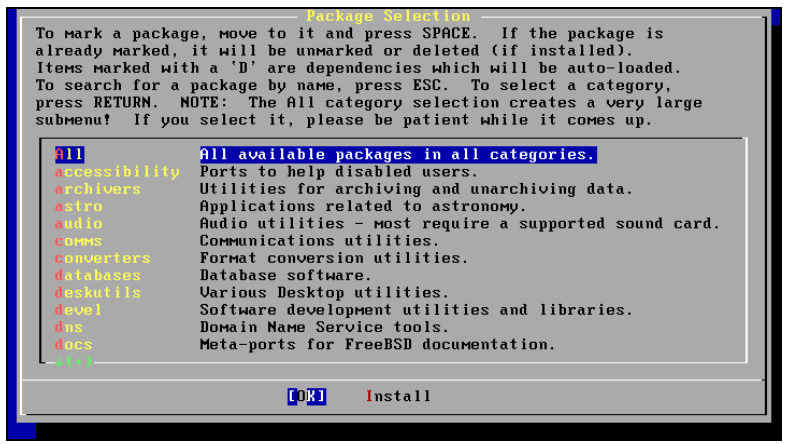


Рис. 18.10. Категории пакетов

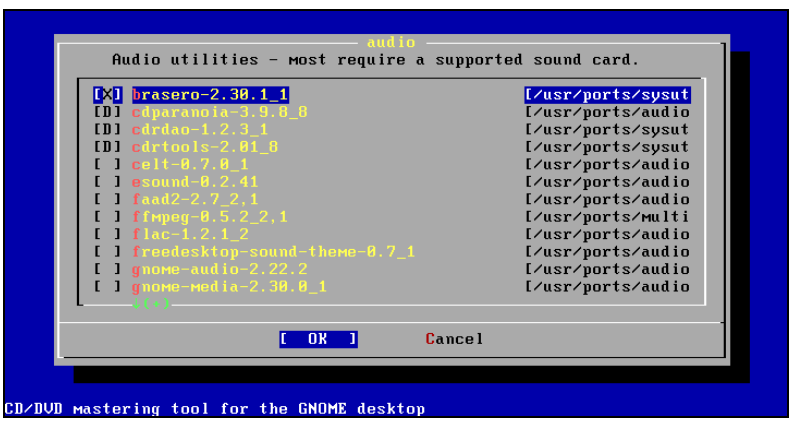


Рис. 18.11. Выбор пакетов

При работе с командой `pkg_info` нужно указывать имя и версию пакета. В моей системе установлен пакет mc, но при вводе команды `pkg_info mc` я получил сообщение о том, что невозможно найти пакет mc (рис. 18.9). Далее с помощью команды `pkg_version` я вычислил точное название пакета (вместе с версией) и получил информацию о нем.

Вот вроде бы и все. Но нельзя не сказать об установке пакетов с помощью программы sysinstall. Некоторым пользователям будет гораздо удобнее устанавливать пакеты с помощью конфигуратора, а не командой `pkg_add`. Запустите sysinstall, выберите меню **Configure**, затем — **Packages**, затем источник пакетов (**CD/DVD**, **FTP** и др.). Сначала вы увидите список категорий пакетов (рис. 18.10). Я зашел в категорию **audio**, и sysinstall отобразил список пакетов из этой категории. Выбрать пакет можно с помощью клавиши <Пробел>. Выделите название пакета и нажмите <Пробел> — возле имени пакета появится крестик (X). При выборе пакета **brasero** для удовлетворения зависимостей (**Dependency**) также будут установлены дополнительные пакеты — они отмечены буквой **D** (рис. 18.11). Так что не удивляйтесь, если вы выберете один пакет, а sysinstall установит еще десяток. Для завершения выбора пакетов в этой категории нажмите **OK** (к этой кнопке можно перейти с помощью клавиши <Tab>). Вы вернетесь опять к списку категорий. Если больше ничего не хотите выбрать, нажмите кнопку **Install.**



Глава 19

Настройка печати

19.1. Системы печати lpr и CUPS

В BSD вы можете использовать одну из систем печати: классическую lpr и более современную CUPS (Common Unix Printing System). "Классика" уже настолько устарела, что я даже и не знаю, как настроить современный принтер в lpr. Все дистрибутивы Linux уже давно забыли про lpr и перешли на CUPS, поэтому я не вижу причин и в FreeBSD пользоваться морально устаревшим программным обеспечением. В этой книге будет рассмотрена настройка современной системы печати CUPS.

19.2. Принтеры и GDI-принтеры

Прежде чем приступить к установке и настройке CUPS, поговорим о самих принтерах. Нет, преимущества и недостатки матричных, струйных и лазерных принтеров мы рассматривать не будем. Но несколько слов сказать все же нужно. Самыми старинными являются принтеры ударного типа (матричные принтеры) — их до сих пор можно встретить в некоторых организациях (банках, школах и т. п.). Как правило, все они подключаются к компьютеру по параллельному порту (LPT). На многих современных компьютерах уже нет такого порта — настолько он устарел. Конечно, выпускаются и новые промышленные принтеры ударного типа, но они применяются в статистических организациях — где нужно много печатать. Современный матричный принтер — это больше исключение из правил, чем правило. В большинстве случаев — это старый принтер, подключаемый к LPT. В базе CUPS драйвер для этого принтера наверняка найдется, а даже если и нет, то можно будет использовать стандартный драйвер.

Совсем другая ситуация с современными струйными и лазерными принтерами. Они подключаются к компьютеру по USB. Поддержка USB в FreeBSD и других BSD-системах имеется, насчет этого можете не беспокоиться. Но некоторые дешевые устройства являются как бы "полупринтерами". Нормальный принтер состоит из "железа" и "математики" — встроенного программного обеспечения. Удешевленные варианты (их еще называют GDI-принтерами) лишены "математики". Управление таким принтером осуществляет специальный драйвер, поставляемый с принтером. Вся беда в том, что такой драйвер существует в большинстве случаев

только для Windows, и заставить такой принтер работать в BSD не получится. А производители GDI-принтеров не торопятся выкладывать исходные тексты драйверов, не говоря уже о разработке драйверов для других операционных систем: FreeBSD или Linux.

Чтобы сэкономить свое время, перед попыткой настройки (а лучше — перед приобретением) принтера убедитесь, что он не является GDI-принтером. Об этом можно прочитать в руководстве по принтеру или найти описание его модели в Интернете.

Далее мы будем считать, что в наличии имеется самый обычный принтер, который может работать как в Windows, так и в любой другой операционной системе.

19.3. Файлы описания принтеров

Вот теперь можно вернуться к CUPS. Преимущества CUPS огромны. Сразу бросается в глаза простая настройка принтера на базе файлов PPD (PostScript Printer Description, файл описания принтера PostScript). Чтобы добавить поддержку того или иного принтера, достаточно добавить в систему необходимый PPD-файл. Найти PPD-файл можно на диске, поставляемом с принтером, или на сайте производителя принтера. В случае необходимости можно поискать его на сайтах **http://www.cups.org/ppd.php** или **http://www.linuxprinting.org**.

Перед настройкой принтера желательно раздобыть файл описания принтера, чтобы не тратить на это время в процессе самой настройки. Для некоторых принтеров файлы описания имеются в портах. Например, для поддержки ряда принтеров производства Dell, Samsung и Xerox можно установить порт splix. Полный список принтеров, поддерживаемых портом splix, представлен в табл. 19.1.

Таблица 19.1. *Принтеры, поддержка которых обеспечивается портом splix*

Производитель	Модель	Состояние
Dell	1100	Работает
Dell	1110	Работает
Samsung	CLP-200	Не тестировался
Samsung	CLP-300	Работает
Samsung	CLP-500	Работает
Samsung	CLP-510	Работает
Samsung	CLP-550	Работает
Samsung	CLP-600	Не тестировался
Samsung	CLP-610	Не тестировался
Samsung	CLX-216X	Работает
Samsung	CLX-2170	Не тестировался
Samsung	CLX-3160	Работает

Таблица 19.1 *(окончание)*

Производитель	Модель	Состояние
Samsung	ML-1510	Работает
Samsung	ML-1520	Работает
Samsung	ML-1610	Работает
Samsung	ML-1630	Работает
Samsung	ML-1640	Работает
Samsung	ML-1710	Работает
Samsung	ML-1740	Работает
Samsung	ML-1750	Работает
Samsung	ML-2010	Работает
Samsung	ML-2150	Не тестировался
Samsung	ML-2250	Работает
Samsung	ML-2251	Работает
Samsung	ML-2510	Работает
Samsung	ML-2570	Работает
Samsung	ML-2550	Работает
Samsung	ML-3050	Не тестировался
Samsung	ML-3560	Работает
Samsung	SCX-4200	Работает
Samsung	SCX-4500	Работает
Xerox	Phaser 3115	Не тестировался
Xerox	Phaser 3116	Не тестировался
Xerox	Phaser 3117	Работает
Xerox	Phaser 3120	Не тестировался
Xerox	Phaser 3121	Работает
Xerox	Phaser 3122	Не тестировался
Xerox	Phaser 3130	Работает
Xerox	Phaser 3150	Не тестировался
Xerox	Phaser 3160	Работает
Xerox	Phaser 3420	Не тестировался
Xerox	Phaser 3425	Не тестировался
Xerox	Phaser 5500	Не тестировался
Xerox	Phaser 6100	Работает
Xerox	Phaser 6110	Работает
Xerox	WorkCentre 3119	Не тестировался

Устанавливается порт splix следующими командами:

```
# cd /usr/ports/print/splix/
# make install clean
```

Для принтеров производства HP следует установить порт hplip, обеспечивающий поддержку принтеров семейств DeskJet, OfficeJet, Photosmart, PSC, Business Inkjet, LaserJet, LaserJet MFP — всего более 1000 моделей. Установка порта hplip осуществляется следующими командами:

```
# cd /usr/ports/print/hplip/
# make install clean
```

Команды установки портов splix и hplip нужно вводить только после установки CUPS, но перед настройкой принтера — чтобы в вашем распоряжении уже были PPD-файлы. Впрочем, в ряде случаев проще найти в Интернете единственный файл, соответствующий модели вашего принтера, а не устанавливать всю базу PPD-файлов ради поддержки одной модели. Если вы все же установили порты splix или hplip, то нужный PPD-файл можно найти в каталоге /usr/local/share/cups/model/<производитель>. Например, для принтеров Samsung PPD-файлы будут помещены в каталог /usr/local/share/cups/model/samsung.

19.4. Установка CUPS

Установим CUPS из портов:

```
# cd /usr/ports/print/cups
# make install clean
```

Можно также воспользоваться утилитой `pkg_add` (если не нужно указывать дополнительные опции при сборке), к тому же `pkg_add` сэкономит время:

```
# pkg_add -r cups
```

Мне хочется свято верить в существование рабочей станции под управлением FreeBSD, но в большинстве случаев — это все-таки сервер. Поэтому кроме порта cups вам еще понадобится порт cups-samba, позволяющий предоставить в общее пользование принтеры, настроенные с помощью CUPS:

```
# cd /usr/ports/print/cups-samba
# make install clean
```

После установки CUPS добавьте следующую строку в /etc/rc.conf:

```
cupsd_enable="YES"
```

Теперь запустим CUPS:

```
# /usr/local/etc/rc.d/cupsd start
Starting cupsd
```

19.5. Установка принтера

Подготовьте PPD-файл принтера, подключите принтер к компьютеру и включите его питание. Настройка CUPS осуществляется через Web-интерфейс. Теоретически, можно управлять принтерами удаленно — с любого компьютера вашей сети.

Но мы пока будем использовать локальный доступ, а потом разберемся, как организовать удаленное управление принтерами. По умолчанию, доступ к Web-интерфейсу разрешается с адреса 127.0.0.1 (локальный компьютер), при этом используется порт 631.

Если графический интерфейс не установлен, придется использовать текстовый браузер, например, links или lynx. Введите команду:

```
# links http://127.0.0.1:631
```

Вы увидите интерфейс управления CUPS, представленный на рис. 19.1.

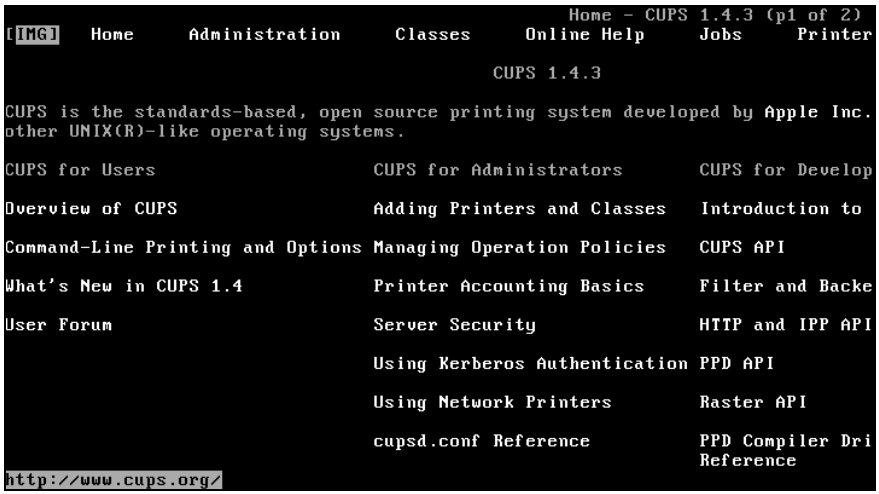


Рис. 19.1. Интерфейс управления CUPS

СОВЕТ

Текстовый браузер links, который мы будем использовать далее, не очень удобен по сравнению с его графическими собратьями, но к нему можно привыкнуть. Стрелки <←> и <→> выполняют функции кнопок **Назад** и **Вперед** соответственно, для выбора ссылки служат стрелки <↑> и <↓>, а для подтверждения выбора нужно нажать клавишу <Enter>.

С помощью стрелок выберите команду **Adding Printers and Classes**, а затем — **Add Printer (**рис. 19.2).

Откроется окно с требованием ввести имя пользователя и пароль (рис. 19.3). Введите имя пользователя root и соответствующий ему пароль. Кстати, если у root нет пароля, то аутентификацию пройти не получится, поэтому переключитесь на другую консоль, войдите как root и с помощью команды `passwd` установите пароль, если это не было сделано ранее.

Теперь можно выбрать добавляемый принтер. В моем случае это **USB Printer #1** (рис. 19.4), в вашем случае, думаю, будет точно такой же вариант (при условии, что к компьютеру подключен всего один принтер). Выберите этот вариант с помощью стрелки <↓>, затем нажмите клавишу <Enter>, чтобы установить флажок слева от **USB Printer #1**, после чего, нажимая стрелку <↓>, выберите кнопку **Continue** и нажмите клавишу <Enter>.

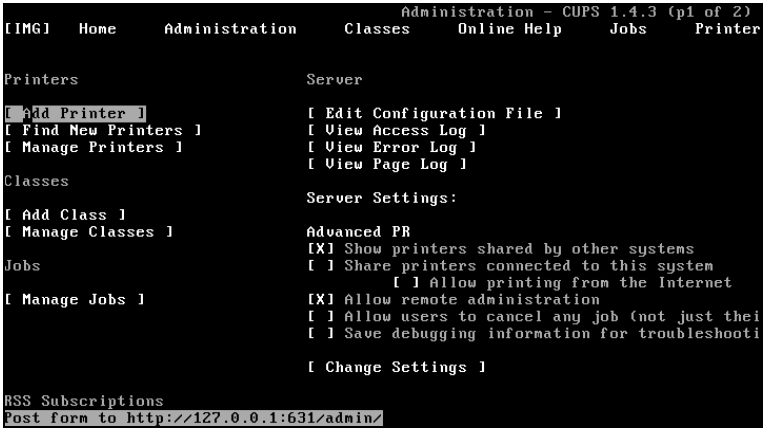


Рис. 19.2. Выберите команду **Add Printer**

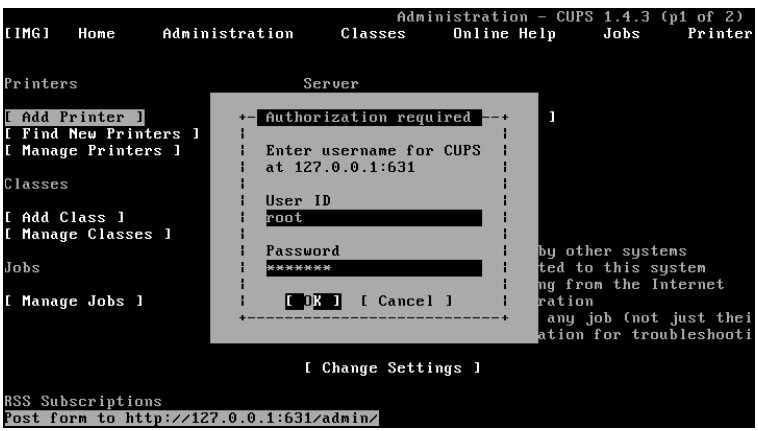


Рис. 19.3. Вводим имя пользователя и пароль root

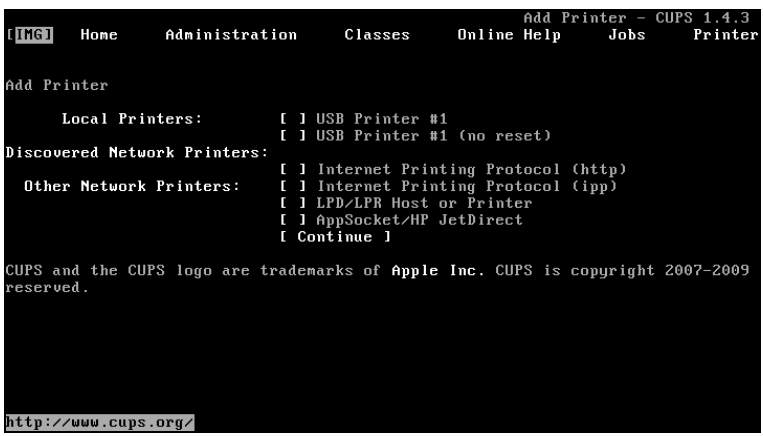


Рис. 19.4. Доступные принтеры

В открывшемся окне (рис. 19.5) введите имя принтера (поле **Name**), его описание (поле **Description**) и месторасположение (поле **Location**) — все это информационные поля, поэтому в них можно вводить произвольный текст (рис. 19.6). Если есть желание сделать принтер общим, выделите параметр **Share This Printer** и нажмите клавишу <Enter> для установки соответствующего флажка. Затем выберите кнопку **Continue** и нажмите клавишу <Enter>.

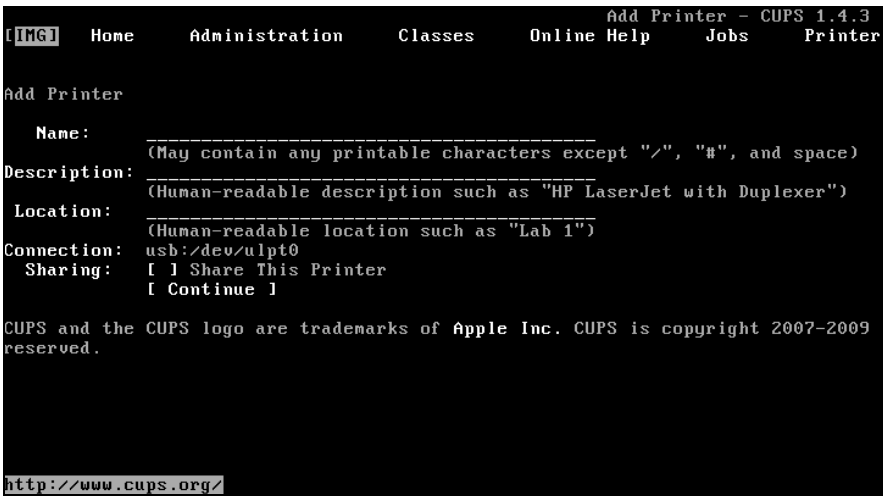


Рис. 19.5. Общие параметры принтера

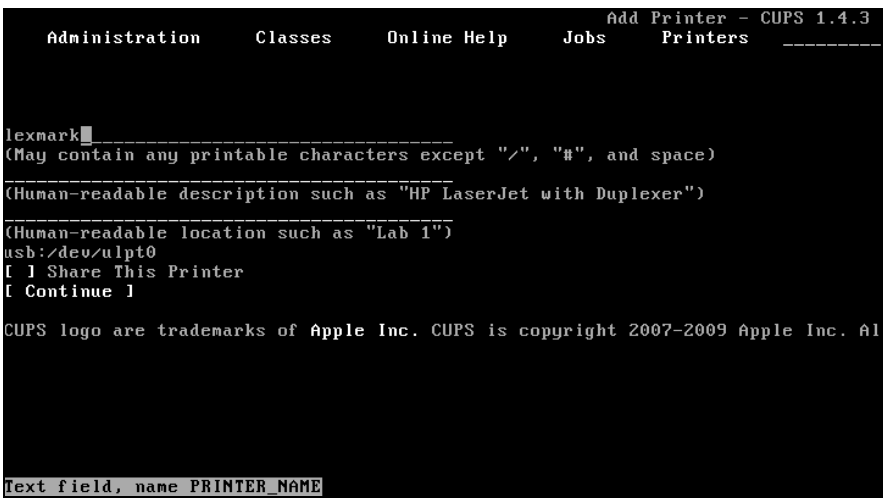


Рис. 19.6. Указано только название принтера — **lexmark**

На следующей странице (рис. 19.7) главное, что нужно сделать — это указать PPD-файл. Для моего старенького принтера Lexmark E321 я нашел PPD-файл в Интернете и выложил на своем сервере: **http://dkws.org.ua/files/lexmark.ppd**. Полу-

чить этот файл можно с помощью утилиты `wget`, которую нужно предварительно установить (рис. 19.8):

```
# pkg_add -r wget
# rehash
# wget http://dkws.org.ua/files/lexmark.ppd
```

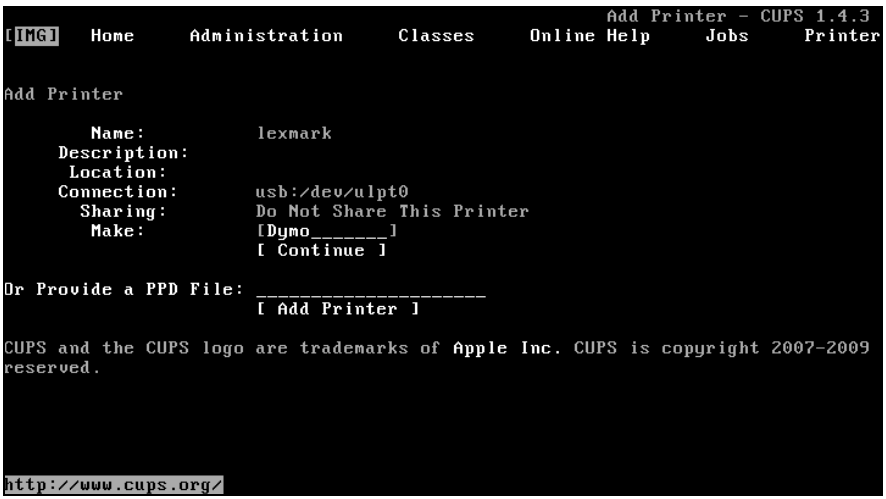


Рис. 19.7. Добавление принтера

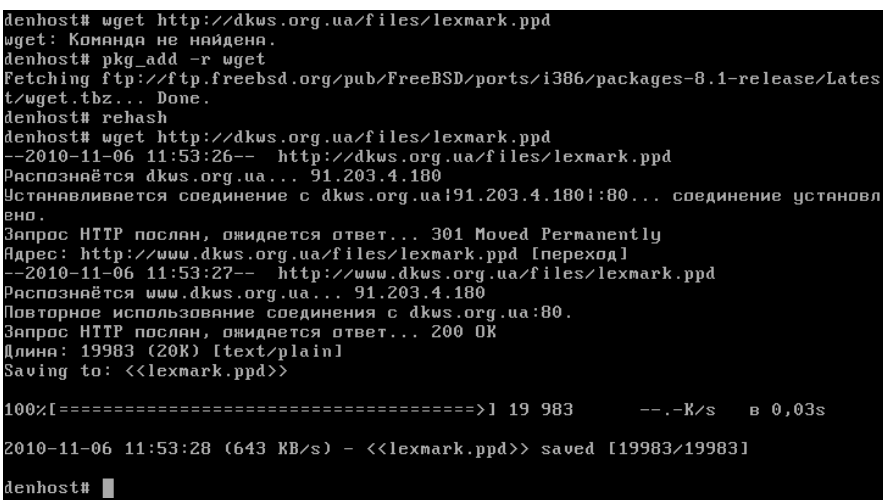


Рис. 19.8. Процесс получения PPD-файла принтера

Нажмите теперь кнопку **Add Printer** (см. рис. 19.7) — вы получите сообщение о том, что принтер добавлен, после чего нужно будет установить параметры по умолчанию (рис. 19.9).

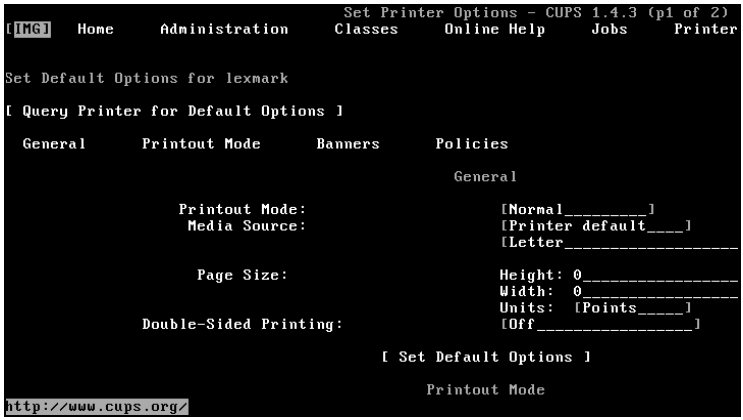


Рис. 19.9. Параметры по умолчанию

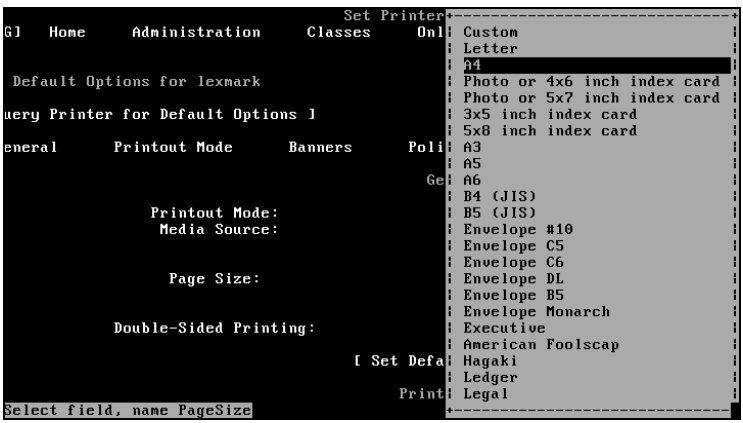


Рис. 19.10. Выбор размера бумаги

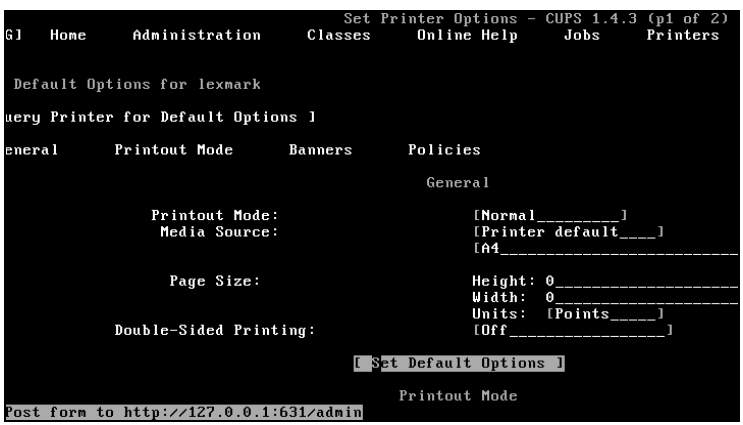


Рис. 19.11. Размер бумаги выбран

Меня устраивали все параметры, кроме формата бумаги (**Media Source**), — по умолчанию предлагается формат **Letter**, но я чаще печатаю на бумаге формата A4 (рис. 19.10, 19.11).

После установки параметров нажмите кнопку **Set Default Options**. Вы увидите сообщение об успешной установке параметров для принтера (рис. 19.12).

Из верхнего меню выберите команду **Printers**, а из предложенного списка принтеров — только что установленный. Откроется страница управления принтером (рис. 19.13) с двумя полезными меню: **Maintenance** (обслуживание) и **Administration** (администрирование). Здесь нужно сделать две вещи: установить наш принтер как принтер по умолчанию и распечатать тестовую страницу.

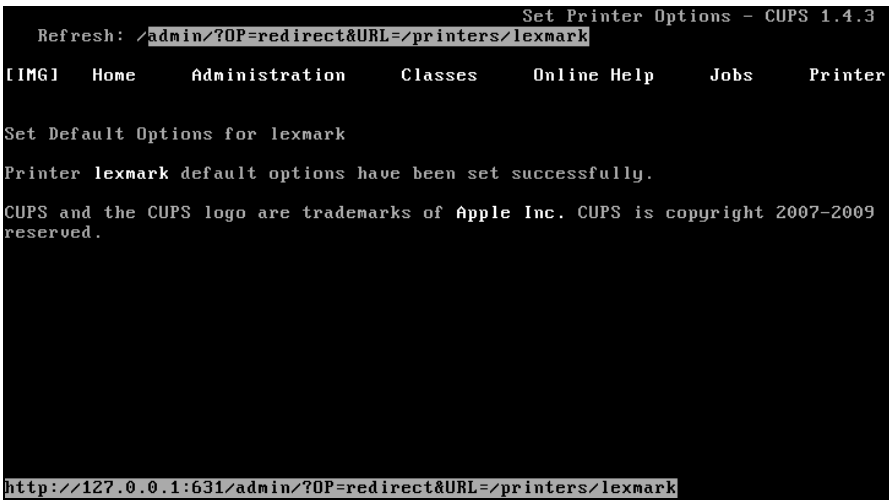


Рис. 19.12. Параметры по умолчанию успешно установлены

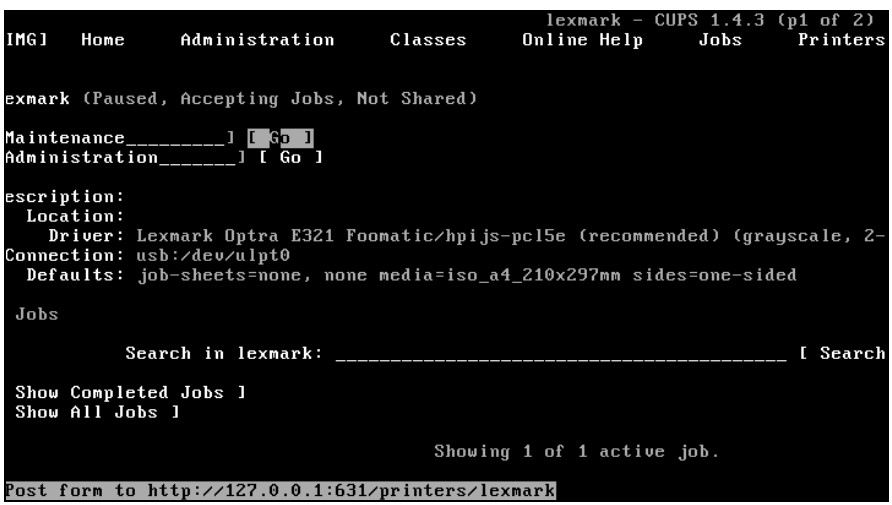


Рис. 19.13. Страница управления принтером

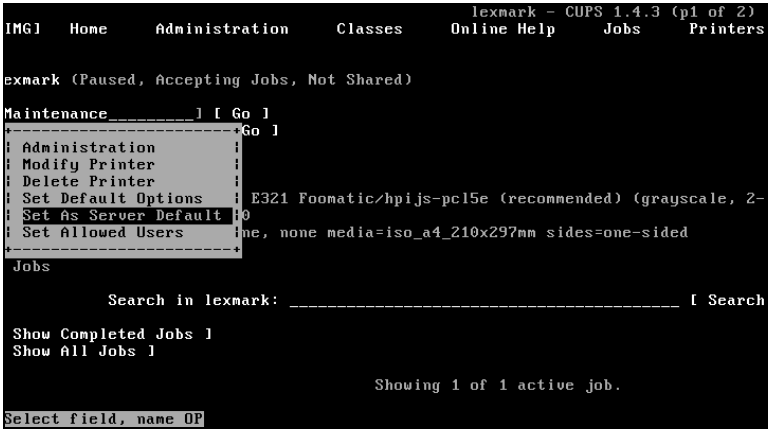


Рис. 19.14. Меню **Administration**



Рис. 19.15. Запуск выбранного действия

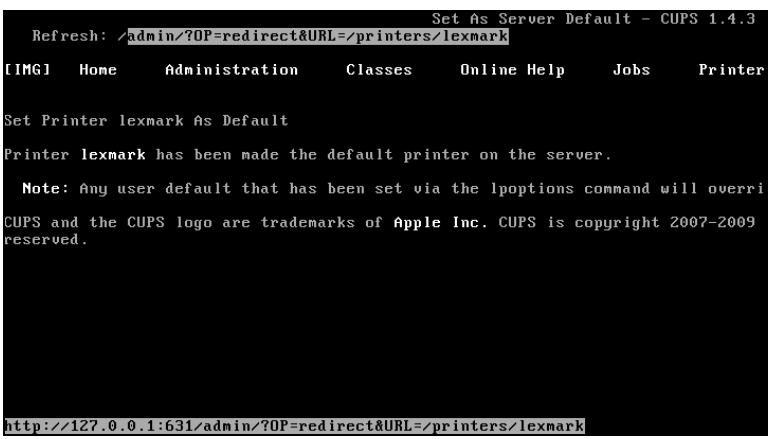


Рис. 19.16. Наш принтер установлен как принтер по умолчанию

Сначала определим принтер по умолчанию — выберите меню **Administration** и нажмите клавишу <Enter>. Выберите в открывшемся меню команду **Set As Server Default** (рис. 19.14). После этого выделите кнопку **Go** у меню **Administration** (рис. 19.15) и нажмите клавишу <Enter>. Вы увидите, что принтер **Lexmark** установлен как принтер по умолчанию (рис. 19.16).

Рассмотрим другие команды меню **Administration** (см. рис. 19.13):

- **Modify Printer** — изменить принтер;
- **Delete Printer** — удалить принтер;
- **Set Default Options** — установить параметры по умолчанию;
- **Set Allowed Users** — установить пользователей, которым разрешено использовать принтер.

Меню **Maintenance** (рис. 19.17) организовано аналогично, в нем есть следующие команды:

- **Print Test Page** — напечатать пробную страницу;
- **Clean Print Heads** — очистить головки (для струйных принтеров);
- **Print Self Test Page** — напечатать страницу самодиагностики. Очень полезная опция, благодаря которой я узнал, что в моем принтере установлено 8 Мбайт памяти, процессор частотой 100 МГц, всего отпечатано 2755 страниц и еще много интересной информации;
- **Resume Printer** — продолжить печать, например, после установки бумаги в лоток;
- **Reject Jobs** — отклонить все задания;
- **Move All Jobs** — переместить задания на другой принтер;
- **Cancel All Jobs** — отметить все задания.

На этом, в целом, настройка принтера завершена. Просмотреть текущие задания печати можно в меню **Jobs** (см. рис. 19.1).



Рис. 19.17. Меню **Maintenance**

Можно также использовать команды `lpr` (печать), `lprm` (удалить задание), `lpq` (просмотреть очередь заданий). Например, сначала мы отправим файл /etc/motd на печать на принтер по умолчанию (если принтер по умолчанию не найден, вы получите соответствующее сообщение), затем просмотрим очередь заданий (команда `lpq`):

```
# cat /etc/motd | /usr/local/bin/lpr
# /usr/local/bin/lpq
```

lexmark готов
Ранг Владелец Задание Файл(ы) Общий размер
1st root 1 (stdin) 1024 байт

Мы увидели, что в очереди есть одно задание — удаляем его командой `lprm`, после чего любуемся пустой очередью заданий:

```
# /usr/local/bin/lprm 1
# /usr/local/bin/lpq
```

lexmark готов
нет записей

ВНИМАНИЕ!

Обычные утилиты `lprm`, `lpq` и `lpr`, находящиеся в каталоге /bin, использовать нельзя, поскольку они ориентированы на старую систему печати lpr. Следует запускать эти утилиты из каталога /usr/local/bin — именно они рассчитаны на систему CUPS. Лучше всего удалить старые утилиты и заменить их символическими ссылками на новые:

```
# cd /usr/bin
# mv lp lp.bck
# mv lpq lpq.bck
# mv lpr lpr.bck
# mv lprm lprm.bck
# ln -s /usr/local/bin/lp /usr/bin/lp
# ln -s /usr/local/bin/lpq /usr/bin/lpq
# ln -s /usr/local/bin/lpr /usr/bin/lpr
# ln -s /usr/local/bin/lprm /usr/bin/lprm
```

19.6. Конфигурационные файлы CUPS

Конфигурационные файлы CUPS находятся в каталоге /usr/local/etc/cups. Все добавленные вами принтеры описаны в файле /usr/local/etc/cups/printers.conf. Обычно вы вряд ли будете редактировать этот файл вручную, если есть довольно удобный Web-интерфейс. Однако иногда может понадобиться изменить некоторые не столь важные параметры, например, описание принтера — чтобы не тратить время на путешествие по дебрям меню панели управления CUPS.

Основным конфигурационным файлом CUPS является /usr/local/etc/cups/cupsd.conf. Этот файл вы тоже вряд ли будете редактировать вручную. Только единственный раз понадобилось отредактировать этот файл — когда я захотел разрешить доступ к панели управления с других компьютеров. По умолчанию файл cupsd прослушивает только локальный интерфейс. Эту опцию обеспечивает строка:

```
Listen 127.0.0.1:631
```

Но вы можете указать IP-адрес своего компьютера, чтобы cupsd прослушивал еще один сетевой интерфейс, добавив в файл cupsd строку:

```
Listen 192.168.1.1:631
```

В результате к панели управления CUPS можно будет обратиться с другого компьютера локальной сети. Для этого введите в браузере на этом компьютере URL **http://192.168.1.1:631**.

На рис. 19.18 показано, как выглядит панель управления CUPS в графическом браузере. Перейдя на вкладку **Принтеры,** мы увидим список настроенных принтеров (рис. 19.19), а выбрав свой принтер, перейдем на страницу управления принтером (рис. 19.20) — обратите внимание на меню **Обслуживание** и **Администрирование.**

Кстати, если вы заметили, панель управления CUPS выводится на русском языке. Удобно? И я так думаю. Почему же я сразу не стал рассматривать панель управления CUPS в графическом режиме? В большинстве случаев вы не будете управлять принтерами с другой машины, потому что обычно в сети работает DHCP-сервер, динамически назначающий IP-адреса компьютерам сети, вследствие чего адреса машин постоянно меняются. Следовательно, вам постоянно придется редактировать директиву `Listen`, что очень не удобно. Конечно, если ваши принтеры подключены к серверу, имеющему статический IP-адрес, то графическая панель управления для вас — просто находка.

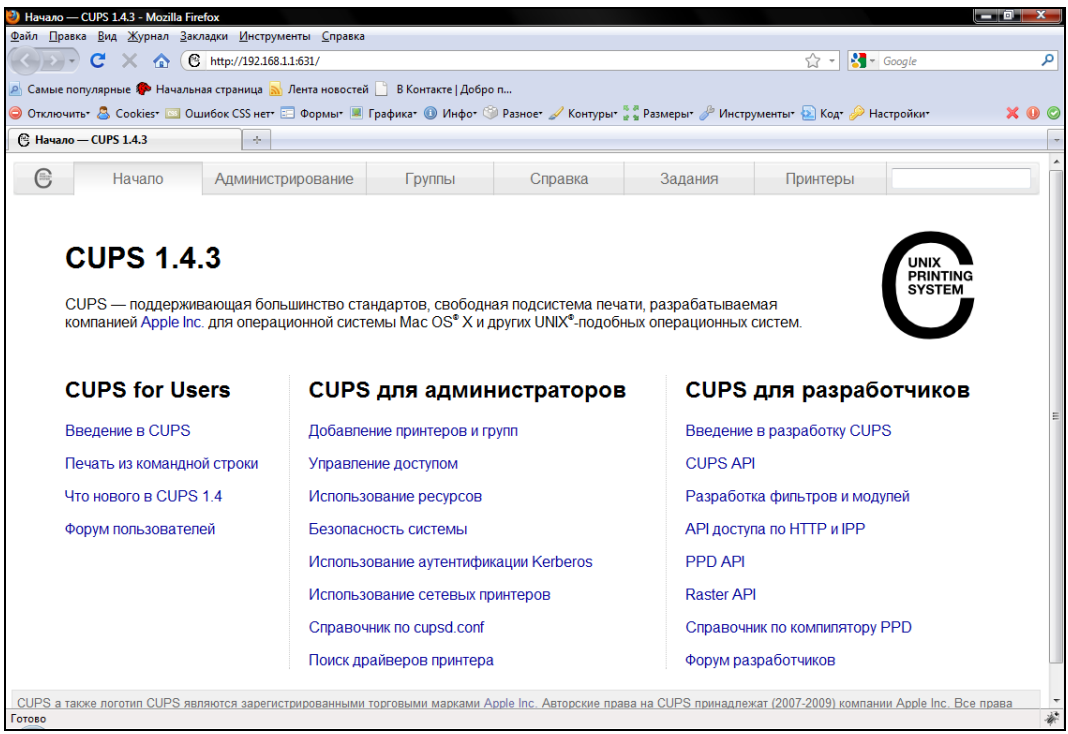


Рис. 19.18. Главная страница панели управления принтерами

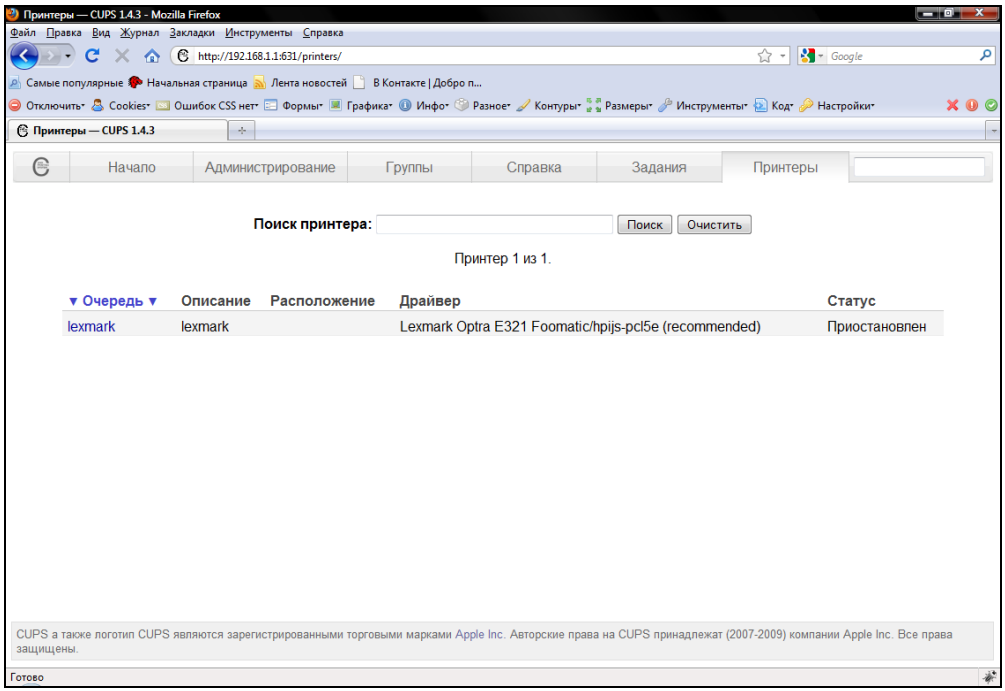


Рис. 19.19. Список настроенных принтеров

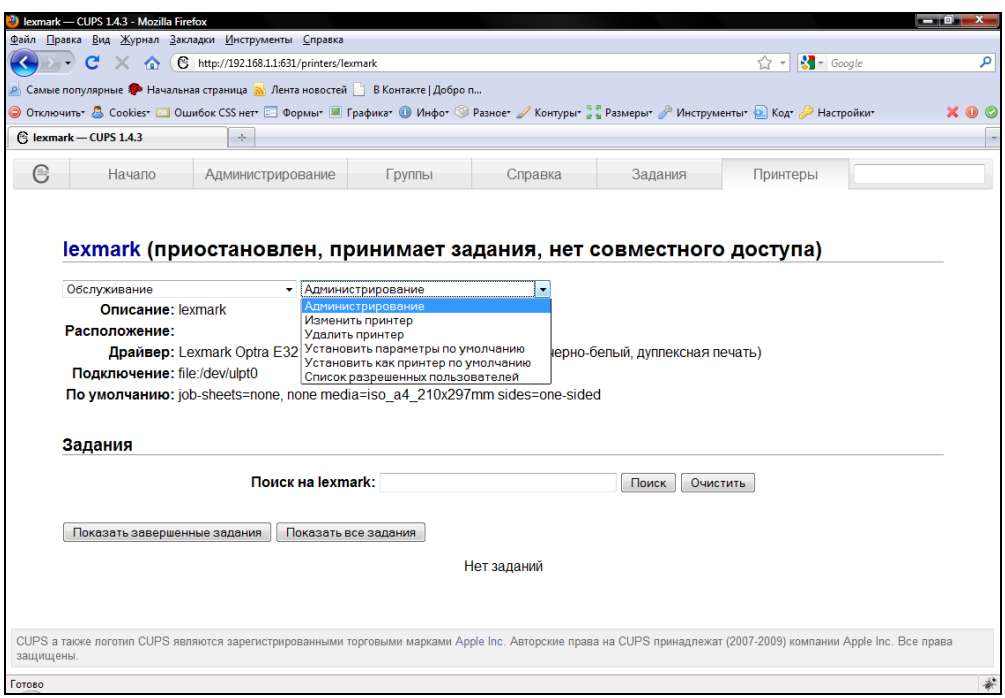


Рис. 19.20. Страница управления принтером

Чуть не забыл. Одной директивы `Listen` — мало. Нужно еще определить, с каких узлов разрешено управлять принтерами. Для этого в файле cupds.conf есть ряд блоков `Limit`, похожих на аналогичные блоки в файле конфигурации сервера Apache. Можно указать IP-адрес компьютера или подсети, а можно указать строку `Allow From All` в каждом блоке `Limit` — все равно для управления принтерами придется вводить пароль root. Пример блока `Limit`:

```
<Location />
    Order Allow,Deny
    Allow From All
</Location>
```



Глава 20

RAID-массивы

20.1. Что такое RAID?

Массивы RAID (Redundant Array of Independent Disk, матрица независимых дисков с избыточностью) обеспечивают более надежное хранение данных. Так, при объединении двух винчестеров в один RAID-массив все, что будет записано на первый винчестер, автоматически продублируется на второй. Если с первым винчестером что-то случится (у жестких дисков есть свойство периодически выходить из строя, это может произойти раз в 5 лет, но, все равно, терять данные не хочется), мы сможем восстановить свои данные со второго винчестера. Приведенный пример является далеко не единственным способом организации RAID-массивов. Алгоритм работы RAID-массива зависит от уровня RAID. Всего существует 6 уровней, описанных в табл. 20.1.

Таблица 20.1. *Уровни RAID-массивов*

Уровень	Алгоритм работы
0	Предназначен не для обеспечения надежности, а для увеличения суммарного объема диска. Предположим, у нас есть два винчестера по 200 Гбайт. Объединив их в один винчестер, мы получим один диск на 400 Гбайт. Очень удобно, если мы работаем с видео (имеется в виду профессиональный видеомонтаж, а не просто просмотр фильмов)
1	Простое зеркальное копирование, как было описано ранее. Все, что записано на первый жесткий диск, будет продублировано на второй. Желательно, чтобы диски были одного размера. Если это не так, то размер RAID-массива будет равен размеру меньшего диска
2	Используется метод битового чередования блоков данных, при этом добавляются коды коррекции ошибок
3	Усовершенствованный уровень 2: коды коррекции ошибок записываются на другой диск
4	Усовершенствованный уровень 3: практически то же самое, но изменен метод записи контрольных кодов

Таблица 20.1 *(окончание)*

Уровень	Алгоритм работы
5	Пока не появился уровень 6, этот уровень был самым надежным. Использует контрольные суммы, и данные записываются вместе с контрольными суммами на все диски. Если с одним из дисков что-то случилось, данные можно восстановить с помощью контрольной суммы. Общий размер массива вычисляется по формуле $M \times (N - 1)$, где N — это количество дисков в массиве, а M — размер наименьшего диска. Минимальное значение $N = 3$
6	Похож на RAID 5, но отличается от него тем, что в каждом ряду данных есть не один, а два блока контрольных сумм, причем эти контрольные суммы многомерные, то есть независимые друг от друга, что позволяет сохранить исходные данные даже при отказе двух дисков в массиве. RAID 6 является более надежным, чем RAID 5, поэтому по возможности нужно использовать RAID 6 (но все же делать резервные копии)

На практике обычно используются уровни 5, 1, 0. Некоторые материнские платы поддерживают RAID-массивы на аппаратном уровне. Раньше поддержкой RAID-массивов обладали только дорогие серверные материнские платы. Сейчас поддержку RAID можно встретить в относительно недорогих материнских платах среднего ценового диапазона. О создании и поддержке аппаратных RAID-массивов вы можете прочитать в документации по вашей материнской плате.

Кроме уровней RAID 1–6, описанных в стандарте, некоторые производители создают комбинированные уровни: RAID 10 (1+0), RAID 15 (1+5), RAID 50 (5+0) и т. д. Суть таких комбинаций заключается в следующем: RAID 10 — комбинация уровней 1 и 0, 15 — уровней 1 и 5 (то есть это зеркало "пятерок") и т. д. Такие комбинированные уровни сочетают в себе преимущества и недостатки своих "родителей". Например, уровень RAID 50 — практически то же, что и RAID 5, но зато быстрее, чем RAID 5.

Кроме обычных уровней, есть еще и расширенные уровни RAID, тогда к наименованию уровня добавляется буква E, например, RAID 1E, RAID 5E и т. д. Это усовершенствованные версии базовых уровней. Чтобы не описывать каждый такой уровень, лучше рассмотрим таблицу с общими характеристиками самых часто используемых уровней RAID (табл. 20.2), то есть именно с характеристиками тех уровней, которые вы будете использовать на практике.

Таблица 20.2. *Характеристики уровней RAID*

Уровень	Избыточность	Мин.	Макс.	Чтение	Запись	Емкость
0	–	1	16	10	10	100
1	+	2	2	8	8	50
5	+	3	16	10	7	67–94
6	+	4	16	10	7	50–88
10 (1+0)	+	4	16	9	9	50

Таблица 20.2 *(окончание)*

Уровень	Избыточность	Мин.	Макс.	Чтение	Запись	Емкость
15	+	6	60	10	7	33–48
1E	+	3	16	8	8	50
1E0	+	2	60	8	8	50
50	+	6	60	10	7	67–94
5E	+	4	16	10	7	50–88
5EE	+	4	16	10	7	50–88
00	–	2	60	10	10	100

Пояснение

Здесь колонка "Избыточность" показывает, поддерживает ли уровень избыточность данных. Колонка "Мин." — минимальное количество дисков в массиве, колонка "Макс." — соответственно, максимальное количество дисков. Колонки "Чтение" и "Запись" — оценки производительности чтения и записи по 10-балльной шкале. Колонка "Емкость" — использование емкости дисков в процентном соотношении.

20.2. Программные RAID-массивы

В FreeBSD можно создавать программные RAID-массивы, даже если ваша материнская плата не поддерживает их на аппаратном уровне. У программных массивов есть один маленький недостаток — они работают немного медленнее аппаратных. Впрочем, у них есть и одно неоспоримое преимущество — поскольку обработка данных происходит на программном уровне, совсем необязательно, чтобы жесткие диски, входящие в состав массива, были совместимы между собой. Например, можно создать массив уровня 5, который будет состоять из дисков EIDE, SATA и SCSI, — это три разных интерфейса, объединить которые в аппаратный массив нельзя никак.

В FreeBSD существует несколько средств настроить программный RAID-массив. В этой книге мы рассмотрим два из них: CCD (Concatenated Disk driver, драйвер объединенного диска) и модульную дисковую структуру (GEOM). Кроме CCD и GEOM можно также использовать менеджер дискового пространства Vinum, рассмотрение которого выходит за рамки этой книги.

20.2.1. Программный RAID-массив на основе CDD

Драйвер CCD позволяет создать RAID уровней 0 и 1, то есть объединить несколько жестких дисков в одну файловую систему или организовать зеркальное копирование данных одного диска на другой. CCD также поддерживает MD-устройства (программный RAID Linux).

Здесь мы создадим с помощью CCD массив RAID уровня 0. Первым делом нужно перекомпилировать ядро *(см. главу 21)*, добавив в файл конфигурации ядра строку:

```
device ccd
```

Загрузившись после перекомпиляции с новым ядром, приступим к настройке дисков. Предположим, что у нас есть четыре диска: ad0, ad1, ad2 и ad3. Первый диск (ad0) будем считать системным — на него установлена FreeBSD. А вот остальные три мы объединим в массив.

Выполните автоматическую разметку трех жестких дисков:

```
# bsdlabel -w ad1 auto
# bsdlabel -w ad2 auto
# bsdlabel -w ad3 auto
```

В результате выполнения этих команд будут созданы разделы ad1c, ad2c, ad3c. Теперь создайте для дисков ad1, ad2 и ad3 разделы e (тип раздела должен быть 4.2BSD):

```
# bsdlabel -e ad1
# bsdlabel -e ad2
# bsdlabel -e ad3
```

После редактирования таблица BSD-разделов будет выглядеть примерно так:

```
c: размер        0    unused        0     0     0   # (Cyl.    0 — N)
e: размер        0    4.2BSD        0     0     0   # (Cyl.    0 — N)
```

Когда указанные операции будут выполнены, введите команду, объединяющую устройства /dev/ad1e, /dev/ad2e, /dev/ad3e в устройство ccd:

```
# ccdconfig ccd0c 32 /dev/ad1e /dev/ad2e /dev/ad3e
```

Здесь: параметр `ccd0c` — это имя устройства /dev/ccd0c (/dev/ можно не указывать), параметр `32` — чередование для файловой системы (см. `man ccdconfig`), параметр `none` — тип массива, который нужно создать (в данном случае `none` означает RAID уровня 0, вместо `none` можно указать `0`), далее следует список устройств, которые нужно объединить в массив.

> ***Примечание***
>
> Если вместо `none` указать `CCDF_MIRROR`, будет включено зеркалирование (то есть организован RAID уровня 1). Параметр `CCDF_LINUX`, указанный вместо `none`, обеспечит поддержку MD-устройств Linux:
>
> ```
> # ccdconfig ccd0 128 CCDF_MIRROR /dev/da4 /dev/da5 /dev/da6
> # ccdconfig ccd0 128 CCDF_LINUX /dev/da8s2 /dev/da9s3
> ```
>
> Поскольку ни зеркалирование, ни массив, совместимый с MD, мы здесь не рассматриваем, эти команды вводить не стоит.

Мы же вернемся к нашей команде `ccdconfig`. После ее запуска устройство ccd0c будет создано, теперь на этом устройстве нужно создать файловую систему:

```
# newfs /dev/ccd0c
```

После чего записать нашу конфигурацию в файл ccd.conf:

```
# ccdconfig -g > /etc/ccd.conf
```

При перезагрузке сценарий /etc/rc (если существует файл /etc/ccd.conf) запустит команду `ccdconfig` и, таким образом, наша конфигурация будет восстановлена.

Если вы загрузились в однопользовательском режиме, не забудьте перед монтированием ccd-устройства ввести команду `ccdconfig`.

Осталось только добавить в /etc/fstab строку:

```
/dev/ccd0c     /mnt/ccd              udf     rw      2       2
```

20.2.2. Программный RAID-массив на основе GEOM

GEOM можно использовать для построения RAID уровней 0, 1 и 3. Для создания RAID уровня 0 служит утилита `gstripe`, для зеркалирования (RAID 1) — утилита `gmirror`, для чередования с контролем четности (RAID 3) — утилита `graid3`. Отличное руководство по GEOM вы можете найти на следующих сайтах:

- **http://www.unixdoc.ru/index.php?mode=2&podmode=1&arcicle_id=169&theme=raid%20freebsd%20geom**;
- **http://www.freebsd.org/doc/ru/books/handbook/geom.html**.

Создание RAID 0

Начнем с уровня 0. Первым делом загрузим модуль geom_stripe.ko:

```
# kldload geom_stripe
```

Чтобы не загружать этот модуль каждый раз при загрузке системы, добавьте в файл /boot/loader.conf строку:

```
geom_stripe_load="YES"
```

Создадим каталог для монтирования нашего RAID-массива:

```
# mkdir /mnt/raid
```

Объединим два неформатированных SCSI-диска в массив st0:

```
# gstripe label -v st0 /dev/da2 /dev/da3
Metadata value stored on /dev/ad2.
Metadata value stored on /dev/ad3.
Done.
```

Командой `bsdlabel` создадим стандартную разметку:

```
# bsdlabel -wB /dev/stripe/st0
```

После этого в каталоге /dev/stripe появятся устройства: st0, st0a, st0c.

На устройстве st0a создадим файловую систему:

```
# newfs -U /dev/stripe/st0a
```

На этом все, логический диск создан и готов к использованию. Смонтируем его:

```
# mount /dev/stripe/st0a /mnt/raid
```

Если все нормально, добавим следующую строку в файл /etc/fstab:

```
/dev/stripe/st0a      /mnt/raid      ufs     rw      2       2
```

Создание RAID 1

Теперь займемся зеркалированием. В общих чертах зеркалирование означает, что диск Б является копий диска А (или же диски В + Г являются копией дисков А + Б) — основная идея заключается в дублировании информации. В случае выхода из строя жесткого диска А, в качестве резервной копии можно использовать диск Б.

Зеркалирование подразумевает использование двух жестких дисков одинаковой емкости.

> ***Внимание!***
>
> Если массив RAID 0 (объединение несколько жестких дисков в одну файловую систему) можно создать из дисков разной емкости, то RAID 1 нужно создавать из дисков одинаковой емкости. Это понятно: если диск А окажется больше диска Б, то будет невозможно создать его полную копию — диск Б заполнится раньше диска А.

Мы подразумеваем, что FreeBSD установлена на жесткий диск, на котором имеется только один слайс — слайс для FreeBSD. Внутри этого слайса есть только два BSD-раздела: один для корневой файловой системы (/), и второй — для раздела подкачки (swap). Можно создать и еще несколько разделов (как было показано в *главе 2*), но тогда придется производить больше манипуляций с командой `bsdlabel`, а для простоты описания это нежелательно.

Прежде чем приступить к настройке, нужно убедиться, что ваше ядро откомпилировано с опцией:

```
options    GEOM_MIRROR
```

В противном случае после всех произведенных действий система не сможет загрузиться с GEOM-устройства.

Введите команду создания устройства /dev/mirror/gm0 и свяжите его с устройством /dev/da1 (это первый SCSI-диск):

```
# gmirror label -vnb round-robin gm0 /dev/da1
```

В этой команде параметр `round-robin` — алгоритм балансировки, который является самым быстродействующим.

Теперь нужно инициализировать GEOM:

```
# gmirror load
```

В результате в каталоге /dev/mirror будут созданы устройства gm0, gm0s1, gm0s1a, gm0s1c. Установим для устройства gm0 стандартную разметку программы fdisk:

```
# fdisk -vBI /dev/mirror/gm0
```

Произведем стандартную разметку командой `bsdlabel`:

```
# bsdlabel -wB /dev/mirror/gm0s1
```

> ***Примечание***
>
> Последние две команды не сработают, только если на диске несколько слайсов и несколько BSD-разделов — придется дополнительно указать их размеры. Для созданной же нами простой конфигурации дисков все должно работать.

Настало время создать файловую систему на устройстве gm0s1a:

```
# newfs -U /dev/mirror/gm0s1a
```

Подмонтируем устройство gm0s1a:

```
# mount /dev/mirror/gm0s1a /mnt
```

Скопируем все данные с нашего загрузочного диска в каталог /mnt:

```
# dump -L -0 -f- / |(cd /mnt && restore -r -v -f-)
```

> ***Совет***
>
> Если на диске имеется несколько файловых систем, то в приведенную здесь команду копирования вместо / нужно подставить имя файловой системы.

Отредактируем файл /mnt/etc/fstab — поскольку наша корневая файловая система будет находиться на устройстве gm0s1a, следует изменить информацию о файловых системах:

```
/dev/mirror/gm0s1a       /       ufs     rw      1       1
```

Теперь на обоих разделах нужно создать файл boot.config с помощью следующих команд:

```
# echo "1:da(1,a)/boot/loader" > /boot.config
# echo "1:da(1,a)/boot/loader" > /mnt/boot.config
```

Таким образом мы сформировали конфигурацию загрузчика на оба раздела для того, чтобы в случае невозможности загрузки с нового раздела (gm0s1a) была возможность загрузки со старого раздела.

В файл /mnt/boot/loader.conf нужно добавить строку, обеспечивающую загрузку модуля geom_mirror.ko:

```
geom_mirror_load="YES"
```

Перезагружаемся: `# reboot`

Система должна загрузиться с gm0s1a. После загрузки добавим диск da0 к устройству gm0:

```
# gmirror configure -a gm0
# gmirror insert gm0 /dev/da0
```

Благодаря параметру `-a` дублирование информации станет осуществляться автоматически.

В итоге получится, что в нашей системе будет две зеркальные копии: одна на /dev/da0, другая — на /dev/da1.



Глава 21

Компиляция ядра

21.1. Установка исходных кодов ядра

Стандартное ядро GENERIC далеко не всегда соответствует нашим требованиям. Возможно, вам захочется включить поддержку большого объема оперативной памяти или какого-нибудь дополнительного устройства, которое не поддерживается по умолчанию. А может, наоборот, вы желаете выключить все опции и поддержку всех устройств, которые не установлены в вашей системе. Такой подход уменьшит размер ядра — идеально для создания встроенных систем, где каждый байт на счету. Вот поэтому мы сейчас и займемся перекомпиляцией ядра.

Первым делом просмотрите содержимое каталога /usr/src (рис. 21.1). Если в нем пусто, значит, нужно установить исходные коды ядра. Вставьте дистрибутивный DVD и выполните команды:

```
# mount /cdrom
# cd /cdrom/8.1-RELEASE/src
# ./install.sh
```

Вторая команда зависит от версии FreeBSD. Лучше всего перейти в каталог /cdrom и уточнить правильное название каталога.

После ввода последней команды начнется установка исходников ядра. Придется немного подождать:

```
Extracting sources into /usr/src...
  Extracting source component: base
  Extracting source component: bin
  Extracting source component: contrib
  Extracting source component: etc
  Extracting source component: games
  Extracting source component: gnu
  Extracting source component: include
  Extracting source component: lib
  Extracting source component: libexec
  ...
```

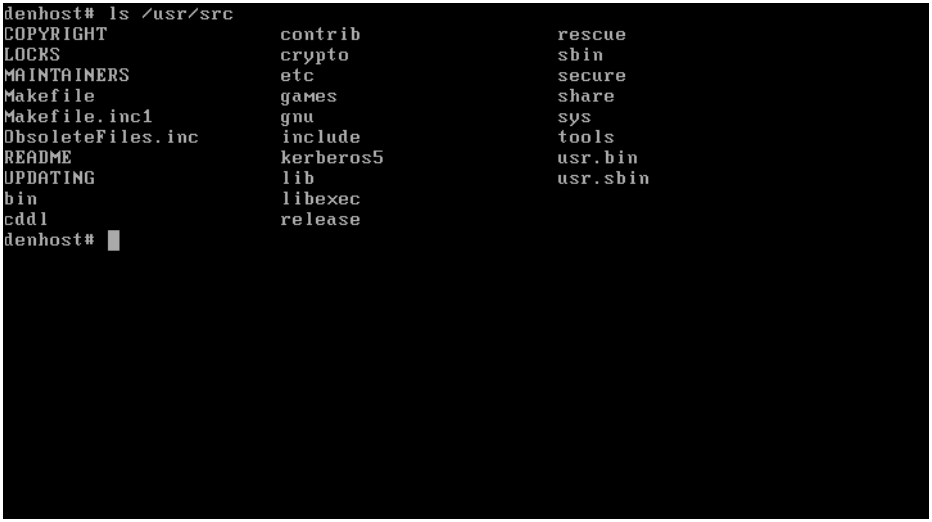


Рис. 21.1. Каталог /usr/src

21.2. Настройка ядра

21.2.1. Архитектура процессора

Прежде чем приступить к сборке ядра, следует определить тип процессора, установленного в вашем компьютере, — уж если собирать собственное ядро, то заодно и оптимизировать его под свой процессор. Ведь ядро GENERIC универсальное, поэтому не использует особенности конкретного процессора, за который вы заплатили деньги. Это все равно, что купить суперкар и ездить на нем со скоростью 20 км/ч или вообще не ездить...

Чтобы узнать, какой процессор установлен в компьютере, введите команду:

```
# dmesg | grep CPU
```

CPU: AMD Athlon(tm)X2 DualCore QL-60 (1899.90 Mhz 686 class CPU)

Как выяснилось, в моем компьютере установлен Athlon X2 — это 64-битный процессор от AMD, поэтому мне нужно перейти в каталог /usr/src/sys/amd64/conf. Если у вас 64-битный процессор от Intel, следует перейти в каталог /usr/src/sys/ia64/conf. Счастливым обладателям SRARC64 — перейти в каталог /usr/src/sys/sparc64/conf. Для сборки ядра для архитектуры i386 — в каталог /usr/src/sys/i386/conf.

21.2.2. Создание копии файла конфигурации ядра

Конфигурация ядра хранится в файле GENERIC. Можно редактировать и этот файл, но лучше создать его копию и работать с копией. Дело в том, что при последующем обновлении исходников ядра файл GENERIC может быть перезаписан — вы потеряете все свои изменения. Так что создайте его копию:

```
# cp GENERIC GENERIC.mykrnl
```

Несмотря на то, что в моем компьютере установлен 64-битный процессор AMD, далее мы будем рассматривать сборку ядра для архитектуры i386, поскольку она более универсальна. Вы же собирайте ядро, которое подходит вашему процессору.

21.2.3. Редактирование файла конфигурации ядра

Откройте в редакторе ее файл `GENERIC.mykrnl`. Если удалить комментарии, то для архитектуры i386 он будет начинаться так:

```
cpu             I486_CPU
cpu             I586_CPU
cpu             I686_CPU
ident           GENERIC

makeoptions     DEBUG=-g

options         SCHED_ULE
```

Сначала указываются архитектуры процессоров, поддерживаемые этим ядром. Далее идет строка идентификации ядра (`GENERIC` — но вы можете указать здесь все, что угодно), затем — опции отладчика (`makeoptions`). После `makeoptions` будут следовать директивы `options` (каждая из которых включает поддержку той или иной функции ядра) и `device` (каждая включает поддержку какого-нибудь устройства).

Что делать дальше? Здесь все зависит от того, какую вы поставили перед собой цель. Если просто желаете оптимизировать ядро под свой процессор, тогда можете приступить к сборке ядра — конфигурационный файл ядра изменять не нужно, достаточно уже того, что вы соберете "родное" ядро. Если же вам нужно включить поддержку какого-то устройства или какой-то опции, то вы должны знать, какую именно строку нужно добавить в файл GENERIC (хотя имя файла можете указать свое — хоть my_super_kernel).

Например, для отключения поддержки шины EISA (уже забыл, когда она была удалена из конфигурации материнских плат), нужно удалить или закомментировать строку:

```
device          eisa
```

Аналогично, можно закомментировать следующую строку для отключения поддержки неактуальных сегодня Floppy-дисководов:

```
device        fdc
```

По умолчанию в код ядра включается поддержка большого количества SCSI-контроллеров:

```
# SCSI-контроллеры
device          ahb             # Семейство EISA AHA1742
device          ahc             # AHA2940 и встроенные AIC7xxx устройства
device          ahd             # AHA39320/29320 и встр. AIC79xx ус-ва
options         AHD_REG_PRETTY_PRINT  # Для отладки
device          amd             # AMD 53C974 (Tekram DC-390(T))
```

```
device        hptiop        # Highpoint RocketRaid 3xxx-серия
device        isp           # Семейство Qlogic
#deviceispfw         # Firmware for QLogic HBAs
device        mpt           # LSI-Logic MPT-Fusion
#devicencr           # NCR/Symbios Logic
device        sym           # NCR/Symbios Logic
device        trm           # Tekram DC395U/UW/F DC315U
device        adv           # Advansys SCSI
device        adw           # Advansys wide SCSI
device        aha           # Adaptec 154x SCSI
device        aic           # Adaptec 15[012]x, 6[23]60
device        bt            # Buslogic/Mylex MultiMaster SCSI
device        ncv           # NCR 53C500
device        nsp           # Workbit Ninja SCSI-3
device        stg           # TMC 18C30/18C50
```

Можно оставить поддержку только необходимых вам контроллеров. Если же у вас вообще нет SCSI-контроллера, закомментируйте все строчки, относящиеся к SCSI.

Вот еще несколько полезных опций ядра:

```
# Автоматическая перезагрузка через 20 сек после kernel panic
options         PANIC_REBOOT_WAIT_TIME=20

# Запрет перезагрузки при нажатии Ctrl+Alt+Del. Для перезагрузки
# используется только команда reboot. Благодаря этому никакой
# "доброжелатель" не подойдет к компьютеру и не нажмет Ctrl+Alt+Del, что
# вызовет его перезагрузку.
# Точнее, подойти-то и нажать он сможет, а вот перезагрузить — нет ☺
# Осталось еще в BIOS SETUP отключить реакцию на кнопку Reset или
# отключить ее аппаратно. Хотя, как уберечься от отключения
# электричества? Розетку на замок не закроешь.
options         SC_DISABLE_REBOOT

# Поддержка растрового текстового режима
options       SC_PIXEL_MODE
options       VESA
# Цвета в консоли будут "зеленый на черном" (ностальгия по Apple),
# а сообщения ядра будут "желтый на черном"
options         SC_NORM_ATTR=(FG_GREEN|BG_BLACK)
options         SC_KERNEL_CONS_ATTR=(FG_YELLOW|BG_BLACK)

# Необходимо для поддержки CR-RW, DVD-RW
device          atapicam
```

По необходимости вы можете включать/выключать те или иные устройства и опции.

21.2.4. Включение PAE — поддержки более 4 Гбайт оперативной памяти

А сейчас я расскажу вам, как включить для процессоров архитектуры i386 (точнее, IA-32) поддержку оперативной памяти объемом более 4 Гбайт.

Поддержка оперативной памяти более 4 Гбайт достигается благодаря включению режима PAE (Physical Address Extension, расширение физического адресного пространства). Впервые расширение физического адресного пространства появилось в процессорах Intel Pentium Pro. Включение PAE позволяет поддерживать до 64 Гбайт оперативной памяти.

Перед включением PAE вам нужно знать следующее:

- поддержка PAE до сих пор не была тщательно протестирована, даже несмотря на ее включение в стабильные версии FreeBSD. Но именно по этой причине она не включается по умолчанию;
- поддержка PAE в FreeBSD существует только для процессоров архитектуры IA-32.

Отсюда можно сделать вывод: если вы хотите использовать больше 4 Гбайт оперативной памяти (скажем, 6 Гбайт), вам нужно использовать или ядро i386 с включенным PAE, или перейти на другую архитектуру (например, на AMD 64). Но переход на другую архитектуру должен быть полным: процессор AMD 64 и ядро, откомпилированное под AMD 64. Если вы установите на компьютер с процессором AMD 64 FreeBSD для архитектуры i386, поддержки больших объемов ОЗУ у вас не будет.

> ***Архитектура процессора***
>
> Процессоры AMD Opteron™, AMD Athlon™ 64, AMD Turion™ 64 и новое поколение AMD Sempron™ относятся к архитектуре AMD 64 (ранее известной как x86-64 или "Hammer"). Процессоры Intel vPro™, Intel Celeron D (некоторые модели, начиная с "Prescott"), Intel Centrino® Duo, Intel Centrino® Pro, Intel Viiv™, Intel Core™2 Extreme, Intel Core™2 Quad, Intel Core™2 Duo, Intel Xeon относятся к архитектуре IA-64 (ранее EM64T).

4 Гбайт на сегодняшний день — тоже не мало. 64 Гбайт, о которых идет речь при включении PAE, — это, скорее, теория, чем практика. На практике вы будете сталкиваться с серверами с объемом оперативной памяти 4–6 Гбайт. Так зачем так много информации о включении PAE, если 4 Гбайт и так поддерживаются? На самом деле, если не включать PAE, то системе будут видны только 3,2 Гбайт (в некоторых случаях 3,6 Гбайт) — то есть ядро i386 без PAE не увидит даже 4 Гбайт. После включения PAE станет доступна вся физическая память — теоретически до 64 Гбайт.

Относительно архитектуры AMD 64 могу сказать следующее: с ней до сих пор не все гладко. Да, установив ядро AMD 64 (или сразу всю FreeBSD, откомпилированную под AMD 64), вы получите поддержку ОЗУ больше 4 Гбайт — 6 Гбайт поддерживаются точно. При этом вы будете использовать все преимущества процессора архитектуры AMD 64 и поддержку ваших честно установленных 4 Гбайт оперативной памяти без всякой перекомпиляции ядра. Но учтите, что далеко не все порты собираются под эту архитектуру, а поэтому вместо одной головной боли

(поддержка 4 Гбайт и более) вы получаете множественную зубную боль в виде проблем с компиляцией многих программ и драйверов.

Если обобщить все здесь сказанное, можно сделать следующее выводы:

- при условии, что ваша система "видит" 3,6 Гбайт из 4 установленных, можно пожертвовать 400 мегабайтами и вообще не обращать внимания на проблему. Никаких неудобств не будет. Если не хватит 3,6 Гбайт памяти, можно сделать раздел подкачки побольше или создать файл подкачки;
- если у вас процессор AMD 64 и установлено 4 Гбайт (или больше) оперативной памяти, скачайте образ диска для архитектуры AMD 64 и установите систему с ядром, откомпилированным под нее. В результате вы получите поддержку всей оперативной памяти, но проблемы со сборкой программ из портов;
- если вы желаете получить стабильно работающую систему с поддержкой более 4 Гбайт оперативной памяти, то *вне зависимости* от типа процессора установите версию FreeBSD для i386 и перекомпилируйте ядро с включенным PAE.

Фанаты и мазохисты могут установить систему i386 без PAE и SMP и превратить ее в систему AMD 64 путем перекомпиляции ядра и мира. Сей процесс рассматривать в книге мы не будем, поскольку обычно в нем нет необходимости. Она может появиться в одном случае — когда сервер удаленный (например, вы арендуете сервер для хостинга), и нет возможности установить на него FreeBSD для AMD 64[1].

> ***Примечание***
>
> Чуть ранее вы увидели незнакомый для себя термин — "мир". Мир в FreeBSD — это базовая система без ядра. По сути, операционная система FreeBSD состоит из ядра и мира — набора базовых (системных) программ, без которых невозможна работа. Если вы не собираетесь радикально переделывать свою систему (например, переходить с платформы i386 на AMD), пересобирать мир надобности нет — достаточно включить необходимые вам опции и пересобрать ядро. Пересборка мира[2] обязательна, когда вы меняете архитектуру своей системы (программы должны быть откомпилированы под целевую архитектуру процессора) и при обновлении системы.

Для сборки ядра с поддержкой PAE нужно в файл конфигурации ядра добавить строку:

```
options PAE
```

Еще одна полезная опция для сервера — SMP (Symmetric Multiprocessing). Для ее поддержки нужно добавить строку (желательно в самое начало):

```
include SMP
```

---

[1] Заинтересовавшиеся могут прочитать статью об удаленной миграции с i386 на AMD 64: **http://www.lissyara.su/articles/freebsd/tuning/migration_from_i386_to_amd64/**.

[2] О пересборке мира можно прочитать в следующих статьях:
http://www.freebsd.org/doc/ru/books/handbook/cutting-edge.html,
http://www.freebsd.org/doc/ru/books/handbook/makeworld.html,
http://c-reaction.net/content/171.

21.3. Сборка ядра

Настал момент истины. Сохраните ваш файл GENERIC.mykrnl. После этого введите команды:

```
# cd /usr/src
# make buildkernel TARGET_ARCH=amd64 KERNCONF=GENERIC.mykrnl
# make installkernel TARGET_ARCH=amd64 KERNCONF=GENERIC.mykrnl
```

Вместо указанной в командах архитектуры `amd64` нужно вписать вашу — например, i386. Вторая команда запускает сборку ядра, третья — устанавливает только что собранное ядро. По завершении третьей команды нужно ввести команду `reboot`.

Если при перезагрузке с новым ядром что-то пойдет не так, тогда нажмите Reset (в случае зависания системы) или просто перезагрузите командой `reboot` (если система реагирует на ваши команды). В загрузочном меню BootMgr (см. рис. 16.2) выберите вариант 6 (**Escape to loader prompt**) — вы увидите командную строку загрузчика. Введите следующие команды:

```
unload
load kernel
boot
```

Эти команды удалят из памяти текущее ядро и загрузят ядро по умолчанию (рис. 21.2). Когда система запустится, повторите компиляцию ядра — возможно, вы допустили ошибку в конфигурационном файле ядра.

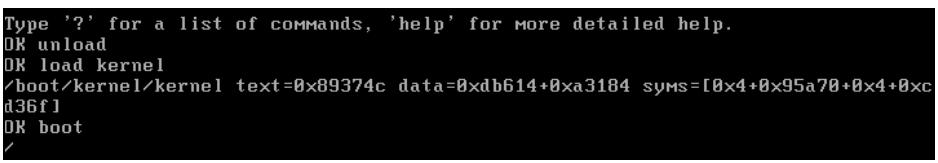


Рис. 21.2. Загрузка ядра по умолчанию



ЧАСТЬ V

Серверное применение BSD



Глава 22

Основы сетевого взаимодействия

22.1. Краткая история сетей

С появлением первых электронно-вычислительных машин (не персональных компьютеров, а именно тех огромных вычислительных машин, которые занимали целые комнаты) возникла проблема переноса данных между ними. С того момента было создано много различных сетей. Сейчас мы вкратце рассмотрим историю сетей, чтобы вы знали, откуда они появились, а потом попробуем классифицировать все имеющиеся виды сетей.

22.1.1. 1941–1975 годы

Первый период развития вычислительных сетей начинается в 1941 году (тогда, если вы помните, появилась первая "большая" ЭВМ) и называется *лабораторным* — в то время сети, как впрочем и ЭВМ, не выходили за пределы лабораторий научных институтов. Тем не менее, с самого начала ставилась задача объединения в сеть ЭВМ без привязки к конкретной аппаратуре.

Любопытно

Казалось бы, как давно это было! Но самое интересное, что мы до сих пор используем решения, разработанные в то время. Последовательный интерфейс RS-232C и параллельный интерфейс LPT/Centronics (да, тот, который служит для подключения принтеров) используются до сих пор. Интерфейс RS-232C постепенно вытесняется современными последовательными интерфейсами: USB и IEEE 1394 (FireWire), и на некоторых современных компьютерах его больше нет вообще. Однако интерфейс Centronics имеется на каждом современном стационарном компьютере, хотя большинство производителей принтеров уже практически перешло на USB. Наличие "старых" интерфейсов зависит только от производителя материнской платы — как он решит, так и будет. Компьютер, на котором я пишу эти строки, был куплен в феврале 2008 года. Тогда я не обратил внимания на наличие/отсутствие старых интерфейсов, но потом выяснилось, что на материнской плате отсутствует RS-232C, но наличествуют Centronics (то есть LPT), а также USB, IEEE 1394, HDMI (правда, он не имеет никакого отношения к сетевым интерфейсам) и другие современные разъемы, которых не было на более старых компьютерах. С другой стороны, в продаже до сих пор можно встретить материнские платы с RS-232C, а также предлагаются отдельные PCI-контроллеры или PC-Card (для ноутбуков), добавляющие два порта RS-232C, если в них возникает острая необходимость.

Интерфейсы RS-232C и Centronics — это, само собой, хорошо, но они годятся только для связи "точка–точка", то есть для непосредственной связи отправителя и получателя данных. Понятно, что в сети может быть гораздо больше, чем две ЭВМ, поэтому разработчики сетей на этом не остановились, и в 1974 году компания IBM представила универсальную архитектуру вычислительных сетей: SNA (System Network Architecture). Эта архитектура, помимо всего прочего, поддерживала *адресацию узлов* сети, смысл которой в том, что каждому узлу сети присваивается уникальный адрес, по которому можно обратиться к этому узлу. Сейчас для адресации узлов преимущественно используются протоколы IPv4 и IPv6, о которых мы поговорим далее в этой главе *(см. разд. 22.5, 22.6)*.

22.1.2. 1976–1982 годы

Второй период развития сетей начался в 1976 году, когда сети вышли за пределы лабораторий, и начали активно разрабатываться сетевые архитектуры и технологии передачи данных. Тогда и появилось семейство протоколов X.25 — протоколов передачи данных в системах с коммутацией пакетов. Разработка протоколов X.25 стала очень важным событием, поскольку до появления Интернета они были единственными протоколами, используемыми для создания глобальных сетей, — именно X.25-сети связывали тогда весь мир в единое целое. Затем на базе X.25 был создан протокол Frame Relay, а на его базе — технология ATM. Подробно рассматривать все производные протоколов X.25 мы не будем, поскольку нас сейчас интересуют только ключевые события в развитии сетей (описание истории появления каждого сетевого протокола займет целую книгу, прочитать которую у вас не хватит терпения). Отмечу только, что Frame Relay, как и ATM, здравствуют и по сей день.

В 1979 году был создан первый модем для персональных (!) компьютеров. Я даже догадываюсь, о чем вы сейчас подумали: какие, мол, персональные компьютеры в 1979 году? Какой модем? Да, Personal Computer (PC) от IBM появился в 1981 году, но это не означает, что до этого не было *персональных компьютеров*. Для работы с первыми ЭВМ обычно требовался целый штат специалистов, а персональный компьютер — это устройство, предназначенное только для одного человека, для одного пользователя. И настоящие персональные компьютеры, отвечающие этому определению, появились еще до 1980 года — это были компьютеры компании Apple. А словосочетание "Personal Computer" — всего лишь название, правда, весьма удачное, продукта компании IBM. IBM первая ввела термин PC, и с того времени все компьютеры со сходной архитектурой команд считаются PC-совместимыми.

А все современные модемы являются Hayes AT-совместимыми, то есть совместимыми с набором AT-команд управления модемом, разработанным компанией Hayes. Первый модем Micromodem II был выпущен этой компанией в 1979 году. Он развивал скорость в 300 бод (бит/с) и предназначался для компьютеров Apple.

Еще в лабораторном периоде были разработаны *системы с произвольным доступом*. Впервые они были использованы в начале 1970-х годов в сети Alohanet, объединяющей Гавайские острова. Сначала эти системы считались бесперспектив-

ными, но в мае 1973 года Боб Меткалф (Bob Metcalf) усовершенствовал метод CSMA, на котором они были основаны. Усовершенствованный метод назвали CSMA/CD (Carrier-Sense Multiple Access with Collision Detection, множественный доступ с контролем несущей и обнаружением коллизий). Боб Меткалф планировал использовать этот метод для совместного доступа к сетевым принтерам, но он позже "перерос" в то, что сейчас называется Ethernet-сетью. Тогда сеть CSMA/CD передавала данные по коаксиальному кабелю (как первые Ethernet-сети) со скоростью 2,94 Мбит/с (для того времени это была значительная скорость), а максимальное расстояние передачи данных составляло 1,5 км. В 1978 году Меткалф зарегистрировал копанию 3Com Corporation (наверное, все мы слышали название этой компании), а в 1982 году выпустил первый в мире серийный Ethernet-адаптер для компьютера Apple.

В 1979 году произошло еще одно важное событие — был организован альянс DIX (DEC, Intel, Xerox), результатом деятельности которого стала в 1980 году разработка стандарта Ethernet.

В 1980 году была разработана *модель взаимосвязи открытых систем* (Open System Interconnect, OSI). Эта модель четко определяет семь уровней, которые обеспечивают передачу данных по сети. Модель OSI сугубо теоретическая, но она лежит в основе всех современных сетей. Мы подробно рассмотрим ее чуть позже *(см. разд. 22.4)*.

22.1.3. 1983–1989 годы

Начиная с 1983 года, в институтах и даже некоторых офисах стали появляться первые локальные сети, связывающие компьютеры толстым коаксиальным кабелем. В то время сетевой адаптер стоил очень дорого (например, для ЭВМ VAX стоимость сетевого адаптера превышала 3 тыс. долларов), поэтому локальную сеть могли себе позволить только самые крупные фирмы. Найти тогда "персоналку" с сетевым адаптером было сложно.

В 1985 году Институтом инженеров по электротехнике и электронике (IEEE) был принят стандарт IEEE 802.3 (10Base-5) — Ethernet-сеть на "толстом" коаксиальном кабеле. В 1989 году был принят стандарт IEEE 802.3a (10Base-2), предусматривающий передачу данных по "тонкому" коаксиальному кабелю. Подробно о стандартах Ethernet мы поговорим чуть позже *(см. разд. 22.8.1)*.

Понятно, что Ethernet-сети — не единственный вид локальной сети. В 1988 году IBM превзошла стандарт Ethernet, представив технологию Token Ring с максимальной скоростью передачи данных в 16 Мбит/с (Ethernet того времени предусматривал передачу данных с максимальной скоростью в 10 Мбит/с).

В 1985 году компания StrataCom начала эксплуатацию первых линий T1 со скоростью передачи данных 1,54 Мбит/с. Чуть позже линии T1 стали доступны крупным компаниям и использовались в качестве магистралей для быстрой передачи данных на большие расстояния.

Индивидуальным пользователям в 1980-х годах сети "особо не светили", поскольку сетевое оборудование продолжало стоить весьма дорого. Так, в 1989 году

компания Arc Electronics представила высокоскоростной модем (19,2 Кбит/с) стоимостью "всего" 3595 долларов. Интересно, что этот модем был относительно дешевле модемов других производителей, которые, к тому же, не обеспечивали заявленной ими скорости.

Кто мог позволить себе сети ISDN, радовался скорости передачи данных в 128 Кбит/с (сети ISDN BRI) или 1,54 Мбит/с (ISDN PRI). О цене говорить не будем — ISDN-сети стоили неприлично дорого.

Технологии — это, конечно, хорошо. Но сетевые адаптеры и прочее сетевое оборудование без программного обеспечения — просто железки. Чтобы компьютер мог работать в сети, нужна сетевая операционная система. В 1980-х годах сеть поддерживали следующие ОС: UNIX (и ее вариации), Novell Netware, Microsoft LAN Manager (оболочка для OS/2, появившаяся в 1987 году).

В 80-х годах прошлого века появились и первые сотовые сети — да, сотовая телефония! Первая система сотовой телефонной связи Nordic Mobile Telephone System (кто помнит — первые "мобилки", появившиеся у нас в 1990-х годах, поддерживали стандарт NMT) была запущена в Дании, Швеции, Финляндии и Норвегии в 1981 (!) году. В 1983 году заработали две сотовые сети в Северной Америке: AURORA-400 и AMPS.

22.1.4. 1990–1995 годы

В 1990 году произошел очередной "переворот" в Ethernet-сетях — был принят стандарт IEEE 802.3i (10Base-T), предусматривающий передачу данных по витой паре 3-й категории со скоростью 10 Мбит/с. Переворот заключался в том, что Ethernet-сети стали:

- *надежнее* — при использовании коаксиального кабеля все компьютеры подключались к общему кабелю, и если этот кабель обрывался, то вся сеть "падала". В случае с витой парой все компьютеры сети подключаются к центральному устройству сети — Ethernet-концентратору. Если происходит обрыв кабеля, ведущего к какому-нибудь узлу сети, этот узел исчезает из сети, но вся сеть продолжает работать;
- *проще в установке* — монтаж витой пары намного проще, чем коаксиального кабеля, особенно, если речь идет о "толстом" коаксиальном кабеле.

Позднее был принят стандарт IEEE 802.1D, в котором было определено понятие *моста* (bridge), и Ethernet-сети наконец-то стало можно делить на сегменты для локализации трафика. Сегментация сети особо важна для крупных сетей — ведь чем больше узлов, тем меньше производительность сети.

Через три года сети того времени стали напоминать современные — в них активно начали использоваться концентраторы и мосты, появились первые коммутаторы и двухуровневые сети. В двухуровневых сетях компьютеры одной рабочей группы (одного отдела компании) объединялись между собой концентратором, а сами рабочие группы (то есть концентраторы рабочих групп) подключались через мосты к общей корпоративной магистрали. В качестве магистрали обычно использовалось оптоволокно (стандарт 10Base-FL или IEEE 802.3j, принятый в 1993 году).

С появлением 10Base-FL на оптоволокне Ethernet-сети выходят за пределы зданий и становятся средством для создания "кампусных" сетей. То есть если раньше Ethernet-сети использовались только для создания локальных сетей, то в 1994–1995 годах стандарт 10Base-FL применялся для связи локальных сетей, находящихся в разных зданиях.

Следующим шагом в создании корпоративных сетей стало изобретение многопортового устройства — центрального коммутатора, в котором были объединены все мосты сети. Такая конфигурация получила название collapsed-backbone ("магистраль в точке"). Примерно в это же время родилось понятие *структурированных кабельных сетей* (СКС).

Понятно, что сети росли, и скорости 10 Мбит/с для магистрали стало недостаточно. На тот момент существовала всего одна "быстрая" технология, обеспечивающая передачу данных по оптоволоконному кабелю со скоростью 100 Мбит/с — FDDI (Fiber Distributed Data Interface, распределенный волоконный интерфейс данных). Но в 1992 году компания Grand Junction начала разработку Ethernet-сети, работающей на скорости 100 Мбит/с, и она была стандартизирована в 1995 году (стандарт IEEE 802.3u, сети 100Base-TX, 100Base-T4 и 100Base-FX). В том же 1995 году компания Grand Junction была поглощена компанией Cisco Systems: закон выживания — выживают лишь сильнейшие. После принятия стандартов 100Base-* спрос на технологию FDDI резко пошел вниз, поскольку Ethernet-сети обеспечивали ту же скорость передачи данных, но были намного дешевле только за счет среды передачи данных — витая пара стоит намного дешевле, чем оптоволокно. А в 1998 году появились Ethernet-сети, передающие данные со скоростью 1 Гбит/с, но об этом позже.

А что же происходило в мире глобальных сетей? В 1990 году компания US Sprint начала предоставлять услуги объединения точек через Frame Relay по всей территории США. Тогда почти все высокоскоростные магистрали переводились на технологию ATM, но для подключения клиентов использовался Frame Relay. Однако в 1994 году компания Bell Atlantic начинает предлагать подключение клиентов по технологии ATM.

Не стоит забывать и об операционных системах. В 1993 году появилась первая действительно сетевая ОС от Microsoft — Windows NT, а в 1995 году — нашумевшая ОС Windows 95.

22.1.5. 1996–1999 годы

В эти годы ничего революционного в магистральных каналах связи не случилось, если не считать появления сервисов гарантирования качества обслуживания (QoS, Quality of Service). Но нас интересуют технологии, более близкие к пользователю. Можно сказать, что в эти годы (1995–1999) завершилась эра развития аналоговых модемов. В 1998 году был принят стандарт V.90, который используется и по сей день (если не считать его небольшого усовершенствования V.92, появившегося в 2000 году). Судя по всему, телефонные модемы отжили свое. Сегодня все больше и больше провайдеров предоставляют высокоскоростной доступ к Интернету, а обычные аналоговые модемы практически уже не используются.

Зарождение высокоскоростного доступа произошло как раз в 1995–1999 годах, когда появились первые кабельные и ADSL-модемы. Кабельные модемы (они передают данные по сетям операторов кабельного телевидения) преимущественно применялись в США. В Европе получили большее распространение ADSL-модемы, использующие для передачи данных обычный телефонный кабель. К сожалению, в те годы в России о таких модемах только слышали, но никто их практически не видел.

В мире локальных сетей в 1998 году появилась технология 1000Base-X, передающая данные со скоростью 1 Гбит/с по оптоволокну, а в 1999 году — технология 1000Base-T, передающая данные со скоростью 1 Гбит/с по витой паре.

22.1.6. 2000 — наше время

Понятно, что развитие сетей не останавливается, а только начинается. Все еще впереди. Лет через десять все современные технологии будут казаться нам такими же "древними", какими сейчас кажутся решения 20-летней давности.

Из интересного в мире Ethernet можно отметить появление в 2003 году технологий передачи данных со скоростью 10 Гбит/с (10GBase-SR, 10GBase-LR, 10GBase-ER, 10GBase-SW, 10GBase-LW, 10GBase-EW) и технологии PLC, обеспечивающей передачу данных по сети электропитания. В 2003 году это казалось странным, но сейчас — вполне нормально.

22.2. Классификация сетей

Сети можно классифицировать по:

- занимаемой территории;
- топологии;
- ведомственной принадлежности;
- скорости передачи данных;
- типу среды передачи данных;
- организации взаимодействия компьютеров.

22.2.1. По занимаемой территории

По занимаемой территории сети могут быть локальными, региональными (они же муниципальные сети) и глобальными:

- *локальные* (LAN, Local Area Network) — сети, занимающие небольшую территорию, например, одну комнату или одно здание;
- *региональные* (MAN, Metropolitan Area Network) — сети, охватывающие город (отсюда другое название — муниципальные) или даже область;
- *глобальные* (WAN, Wide Area Network) — такие сети охватывают территории одного или нескольких государств или даже весь мир. Пример всемирной сети — Интернет.

С локальными и глобальными сетями все понятно, разберемся с сетями региональными. Сеть MAN, как правило, объединяет в единое целое несколько сетей — напри-

мер, сети двух или более зданий. При этом среда передачи данных сети MAN может быть как проводной, так и беспроводной.

Беспроводная сеть обходится намного дешевле, чем сеть на базе оптоволокна, но она менее надежна и менее безопасна. И все же беспроводные технологии очень полезны для MAN — не всегда есть возможность проложить кабель. С другой стороны, MAN часто выступает в качестве магистральной сети, поэтому производительности беспроводной сети может оказаться недостаточно.

Сейчас особой необходимости в MAN-сетях нет, поскольку можно организовать *виртуальную частную сеть* (VPN, Virtual Private Network), использующую каналы Интернета для передачи данных. Представим следующую ситуацию: некая организация с главным офисом в Москве открыла свой филиал в Санкт-Петербурге. Как объединить сети офисов вместе? Вы только представьте себе, сколько кабеля для этого понадобится! Причем витой парой здесь не отделаешься, придется использовать дорогой оптоволоконный кабель — ведь расстояние-то большое. Беспроводные технологии тоже из-за расстояния отпадают. Остается только одно — использовать для передачи данных каналы Интернета. Сеть каждого офиса подключается к Интернету через канал местного интернет-провайдера, и через Интернет создается виртуальная частная сеть. И дешево, и быстро — ведь высокоскоростное подключение к Интернету в настоящее время вполне доступно. Понятно, что данные будут передаваться по незащищенным каналам, поэтому в виртуальной частной сети используется шифрование всех передаваемых данных. Механизмы VPN позволяют не только объединить две разные сети в единое целое, но и обеспечить безопасность передаваемых данных.

22.2.2. По топологии

Существуют следующие топологии сети:

- *линейная* (рис. 22.1) — подключение по принципу гирлянды: каждый узел сети подключается к следующему узлу сети. В такой сети от узла с номером 1 до узла N будет всегда одинаковый маршрут: через узлы 2, 3, 4, …, N – 1. Понятно, в случае отказа одного из узлов сети, линейная сеть прекратит свое существование. В настоящее время линейные сети практически не используются (если не принимать во внимание нуль-модемное соединение двух компьютеров между собой);

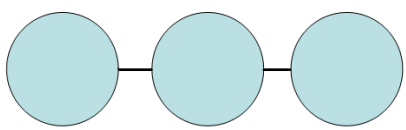


Рис. 22.1. Линейная топология

- *кольцевая* (рис. 22.2) — каждый узел сети соединен с двумя соседними узлами, все узлы сети образуют кольцо. Кольцевая топология используется технологиями Token Ring, FDDI и некоторыми другими;

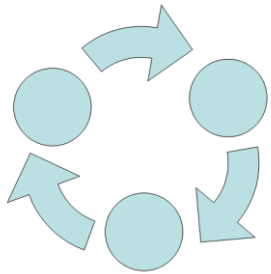


Рис. 22.2. Кольцевая топология

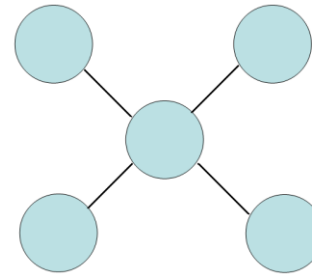


Рис. 22.3. Звезда

- *звездообразная* (рис. 22.3) — в такой сети есть один центральный узел, с которым связан каждый узел сети. Такие сети еще называются *централизованными*. "Падение" центрального узла означает "падение" всей сети. Обычно в качестве центрального узла используется концентратор (hub) или коммутатор (switch). Пример звездообразной сети — Ethernet на базе витой пары;
- *общая шина* (рис. 22.4) — все узлы сети подключаются к единой среде передачи данных, например, к коаксиальному кабелю. Слабое место такой сети — сама среда передачи данных: обрыв кабеля означает сбой всей сети. Пример сети на общей шине — Ethernet на базе коаксиала;
- *древовидная* (рис. 22.5) — топологию этой сети проще представить, чем описать или вникать в определение. В древовидной сети есть более двух конечных узлов и, по крайней мере, два промежуточных узла. В древовидной сети между двумя узлами есть только один путь. Чтобы вникнуть в правильное определение древовидной сети, нужно знать теорию графов, поскольку древовидная сеть — это неориентированный ациклический граф, не содержащий замкнутых путей и позволяющий соединить единственным образом пару узлов;

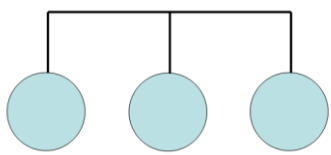


Рис. 22.4. Общая шина

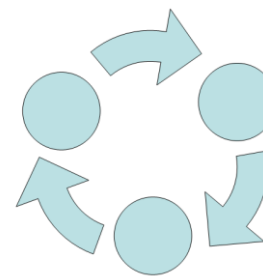


Рис. 22.5. Древовидная топология

- *ячеистая* (рис. 22.6) — в такой сети есть, по крайней мере, два узла, имеющих два или более пути между ними;
- *полносвязная* (рис. 22.7) — сеть, в которой есть связь между любыми двумя узлами. Это самая надежная топология сети, но она практически никогда не используется, поскольку является самой дорогой и труднообслуживаемой.

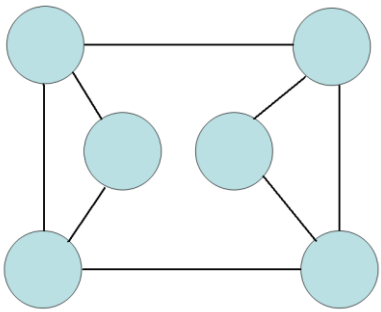


Рис. 22.6. Ячеистая топология

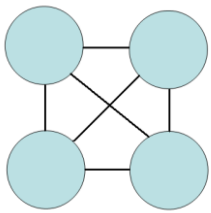


Рис. 22.7. Полносвязная топология

22.2.3. По ведомственной принадлежности

По ведомственной принадлежности различают следующие виды сетей:

- *ведомственные* — принадлежат какой-то организации и находятся на ее территории;
- государственные — используются в госструктурах.

22.2.4. По скорости передачи данных

По скорости передачи данных сети делятся на низко-, средне- и высокоскоростные. Основной критерий разделения — скорость передачи данных. Понятно, что скорость передачи данных — понятие непостоянное. То, что сегодня считается среднескоростным соединением, завтра будет отнесено к низкоскоростным. Тем не менее, сегодня низкоскоростной сетью считается сеть со скоростью передачи информации до 10 Мбит/с. Среднескоростная сеть передает данные со скоростью до 100 Мбит/с, а высокоскоростные сети передают информацию со скоростью свыше 100 Мбит/с.

22.2.5. По типу среды передачи данных

Казалось бы, тут все просто: сети бывают проводными или беспроводными. Но очень часто "в природе" встречаются *смешанные*, или *гибридные* сети, сочетающие как проводные, так и беспроводные технологии. В этой книге больше внимания будет уделяться именно таким сетям. Представьте, что у вас дома есть компьютер и ADSL-модем. Вы купили ноутбук. Подключать ноутбук по Ethernet-кабелю не очень-то хочется — ноутбук по своей природе мобильное устройство, и хотелось бы использовать его по всей квартире (а если у вас свой дом, то и во дворе). Поэтому вам понадобится *точка доступа*, которую вы подключите к существующей сети. Точка доступа в данном случае будет выполнять функцию моста между беспроводной и проводной сетью. Именно с ее помощью ваш ноутбук сможет подключиться к Интернету. Поэтому у вас дома появится смешанная сеть, созданная своими руками.

22.2.6. По способу организации взаимодействия компьютеров

Сети бывают одноранговыми и типа клиент/сервер. В *одноранговой* сети нет выделенного сервера: каждый клиент сети может выступать как в роли сервера, то есть предоставлять услуги другим узлам сети, так и в роли клиента, то есть пользоваться услугами, которые предоставляют другие узлы сети.

В сети *клиент/сервер* есть выделенный сервер, предоставляющий определенные сетевые услуги (какие именно, зависит от самой сети).

22.3. Способы передачи данных в сетях

Любая сеть данных должна использовать какой-нибудь метод коммутации своих абонентов, то есть сеть должна знать, как отправить данные тому или иному компьютеру. В современных сетях распространены три основных метода коммутации: коммутация каналов, коммутация сообщений и коммутация пакетов.

Коммутация каналов используется в аналоговых (нецифровых) телефонных сетях. Для передачи компьютерных данных используется *коммутация пакетов*. Разница между этими методами просто огромна. В первом случае (коммутация каналов) для передачи данных нужен физический канал между двумя узлами. Понятно, что прокладывать кабель между каждой парой узлов сети (между каждой парой телефонов) экономически нецелесообразно, поэтому были созданы *коммутаторы* (сейчас мы говорим о телефонных коммутаторах), соединяющие между собой двух разных абонентов сети по их вызову.

В компьютерных сетях такой способ коммутации совершенно не годится, поскольку канал большую часть времени будет простаивать и без пользы занимать ресурсы коммутатора. Кроме того, при отправке большого количества информации по такой сети на коммутатор ляжет огромная нагрузка, поскольку данные будут переданы за один раз.

Метод коммутации пакетов заключается в том, что передаваемые данные разбиваются на части — *пакеты*. Каждый пакет отправляется отдельно, и, что интересно, два разных пакета, отправленные одним отправителем, могут дойти к получателю разными маршрутами. Например, вы отправляете пакеты компьютеру, не принадлежащему вашей сети. Сначала пакеты отправятся провайдеру, затем — какому-нибудь маршрутизатору Интернета, но если этот маршрутизатор окажется недоступен (мало ли что может случиться), автоматически будет задействован резервный канал, отправляющий данные через другой маршрутизатор. В итоге получится, что первый пакет будет доставлен одним маршрутом, а второй — другим. Однако оба пакета будут доставлены получателю.

К тому же метод коммутации пакетов позволяет использовать физически одну и ту же среду передачи данных (читайте — один и тот же кабель) для (почти) одновременной отправки данных несколькими компьютерами. Рассмотрим ситуацию: у вас в квартире установлено два параллельных телефонных аппарата, и вы разговариваете по одному из них. Если кто-то поднимет трубку второго телефона, то не

сможет набрать номер, поскольку среда передачи информации (телефонный кабель) занята.

В случае с коммутацией пакетов такого нет — в сетях с архитектурой "общая шина" (Ethernet) данные отправляются почти одновременно. Например, компьютерам А и Б нужно отправить данные. Допустим, первым получил доступ к общей среде компьютер А. Он отправляет пакет фиксированного размера. Пока компьютер А отправляет пакет, компьютер Б ожидает доступ к среде. После отправки пакета компьютером А компьютер Б сможет получить доступ к общей среде и отправить свой пакет. Компьютер А в это время делает небольшую паузу. Потом компьютер Б должен сделать паузу, за время которой компьютер А успеет передать следующий пакет. Сами понимаете, время ожидания настолько мизерно, что пользователям компьютеров А и Б кажется, что компьютеры отправляют данные одновременно. Если бы в компьютерной сети использовался метод коммутации каналов, то компьютер Б должен был ждать, пока компьютер А не передаст все данные.

Метод *коммутации сообщений* в чистом виде практически нигде не используется, но он послужил прототипом для метода коммутации пакетов.

22.4. Модель OSI

В 80-х годах прошлого века появилась необходимость стандартизировать различные сетевые технологии. Ведь без стандартизации в мире компьютерных сетей воцарился бы хаос: оборудование различных производителей не смогло бы работать вместе. Поэтому международная организация по стандартизации (International Organization for Standardization, IOS) разработала *модель взаимодействия открытых систем* (Open System Interconnection, OSI). Вы также можете встретить другие названия этой модели: *семиуровневая модель OSI*, или просто *модель OSI*. Эта модель предусматривает семь уровней взаимодействия систем:

1. Физический.
2. Канальный.
3. Сетевой.
4. Транспортный.
5. Сеансовый.
6. Представительный.
7. Прикладной.

Зачем нужна такая система? Предположим, что нам необходимо заставить вместе работать две сети, использующие разную среду передачи данных, — например, витую пару и радиоволны (беспроводную сеть). Если бы не было модели OSI, то для каждой сети пришлось бы разрабатывать модель взаимодействия, а потом придумывать способ, позволяющий заставить две разные по своей архитектуре сети работать вместе. В случае с моделью OSI не нужно "изобретать велосипед" заново. Следует взять за основу уже имеющуюся сеть (в данном случае Ethernet) и переписать физический уровень. В итоге нам не придется разрабатывать браузеры, почтовые клиенты и другие сетевые приложения для каждой сети — браузеру все равно,

какая среда передачи данных используется. Как видите, модель OSI хоть и теоретическая, зато очень полезная. Рассмотрим ее уровни:

- на **физическом уровне** определяются характеристики электрических сигналов, то есть описывается физическая среда данных. На этом уровне и происходит физическая передача данных по кабелю или радиоволнам (в зависимости от типа сети). Пример протокола физического уровня: 1000Base-T — спецификация Ethernet, передающая данные по витой паре 5-й категории (о категориях витой пары см. в *разд. 22.8.3*) со скоростью 1000 Мбит/с;
- **канальный уровень** используется для передачи данных между компьютерами (и другими устройствами, например, сетевыми принтерами) одной сети. Пример протокола канального уровня: PPP (Point-to-Point Protocol). Топология сети (шина, звезда и т. д.) определяется как раз на канальном уровне (в *разд. 22.2.2* мы подробно рассмотрели существующие топологии сетей). На канальном уровне все передаваемые данные разбиваются на части, называемые *кадрами* (frames). Канальный уровень передает кадры физическому уровню, а тот, в свою очередь, отправляет их в сеть.

  На канальном уровне вводится понятие MAC-адреса. *MAC-адрес* — это уникальный аппаратный адрес сетевого устройства (например, сетевого адаптера, точки доступа). Каждому производителю сетевых устройств выделяется свой диапазон MAC-адресов. В мире нет двух сетевых устройств с одинаковыми MAC-адресами;
- теперь рассмотрим **сетевой уровень**. Он используется для объединения нескольких сетей, то есть для организации межсетевого взаимодействия, — ведь протоколы канального уровня могут работать только в пределах одной сети. Канальный уровень не может передать кадр компьютеру, который находится в другой сети. Понятно, что если бы у нас был только канальный уровень и не было сетевого, мы не смогли бы передавать данные между двумя сетями, например, между локальной сетью и Интернетом. Пример протокола сетевого уровня: IP (Internet Protocol). Конечно, IP — это не единственный протокол сетевого уровня, но в этой книге мы будем рассматривать только TCP/IP-сети, поэтому нет смысла упоминать другие протоколы.

  При всем своем желании мы не можем построить огромную сеть, охватывающую весь мир (даже если бы это и удалось, не думаю, что такая сеть работала бы быстро). Поэтому Интернет состоит из совокупности различных сетей, которые объединяются в одно целое маршрутизаторами. Расстояние между сетями исчисляется в количестве переходов пакетов (на сетевом уровне передаваемые данные разбиваются именно на пакеты) через маршрутизаторы. Единица такого перехода называется *хопом* (от англ. hop). Количество хопов равно количеству маршрутизаторов между двумя сетями. Например, от моего узла до узла **volia.net** 6 хопов (рис. 2.8);
- **транспортный уровень** отвечает за доставку пакетов получателю. Не секрет, что при передаче по каналам связи данные могут быть повреждены или вовсе потеряны. Гарантирует доставку пакета в первозданном виде именно транспортный уровень. На этом уровне осуществляются обнаружение и исправление

ошибок, восстановление прерванной связи и некоторые дополнительные сервисы вроде срочной доставки (приоритет пакета) и мультиплексирование нескольких соединений. Самым известным и распространенным протоколом транспортного уровня является TCP (Transport Control Protocol);

```
denis@denis-desktop:~$ traceroute volia.net
traceroute to volia.net (82.144.192.47), 30 hops max, 40 byte packets
 1  router.shtorm.net (195.62.14.2)  0.330 ms  0.306 ms  0.295 ms
 2  border.shtorm.com (195.62.14.7)  0.523 ms  0.515 ms  0.504 ms
 3  194.44.13.13 (194.44.13.13)  9.435 ms  9.426 ms  9.415 ms
 4  volia-10G-gw.ix.net.ua (195.35.65.221)  9.618 ms  9.613 ms  9.603 ms
 5  v109.TenGig3-8.diamond.volia.net (82.144.193.192)  9.358 ms  9.562 ms  9.554
 ms
 6  tower.volia.net (82.144.192.47)  9.538 ms  9.251 ms  9.240 ms
denis@denis-desktop:~$
```

Рис. 22.8. Количество переходов от моего узла до узла **volia.net**

- **сеансовый уровень** отвечает за установку и за разрыв соединения между компьютерами. На этом уровне также предоставляются средства синхронизации. Сеанс сетевого уровня заключается в установке соединения (при установке стороны, между которыми будут передаваться данные, могут договариваться о дополнительных параметрах связи, например, обмениваться ключами), передаче информации и разрыве соединения.

  Многие часто путают сеансы сетевого уровня и сеанс связи. Вы можете установить сеанс связи (например, подключиться к Интернету), но не устанавливать логического соединения, то есть не запустить браузер для соединения с удаленным узлом;
- как было отмечено чуть ранее, на сеансовом уровне стороны могут договориться о дополнительных параметрах, были также упомянуты *ключи*. Однако само шифрование и дешифрование данных осуществляется **представительным уровнем**. Пример протокола этого уровня — SSL (Secure Socket Layer);
- последний (самый высокий) уровень — **прикладной**. На этом уровне работает множество разных протоколов, например, HTTP (Hyper Text Transfer Protocol), FTP (File Transfer Protocol), SMTP (Simple Mail Transfer Protocol) и т. д.

22.5. Что такое протокол?

В этой главе довольно часто упоминалось слово "протокол". *Протокол* — это правила, определяющие взаимодействие компьютеров в вычислительной сети. Рассмотрим несколько самых важных протоколов.

В основе Интернета лежит протокол TCP/IP (Transmission Control Protocol/Internet Protocol). Чтобы система смогла работать в Интернете, она должна поддерживать протокол TCP/IP. Вообще говоря, TCP/IP — это совокупность двух протоколов. Протокол TCP, как уже было отмечено ранее, отвечает за корректность передачи данных по Интернету (точнее, по любой сети, использующей этот прото-

кол), то есть гарантирует доставку данных по сети. Протокол IP используется для адресации компьютеров сети. Дело здесь в том, что у каждого компьютера сети имеется свой уникальный адрес (IP-адрес), и, чтобы передать данные компьютеру, нужно его IP-адрес знать. В *разд. 22.6* и *22.7* мы поговорим о системе доменных имен (DNS, Domain Name System).

Кроме протоколов TCP и IP посетители Интернета работают с теми или иными серверами, использующими следующие протоколы:

- HTTP (Hyper Text Transfer Protocol) — протокол передачи гипертекста. Все Web-серверы Интернета используют протокол HTTP или его безопасную версию HTTPS (HTTP Secure);
- FTP (File Transfer Protocol) — протокол передачи файлов. Используется для обмена файлами между компьютерами. Вы можете подключиться к FTP-серверу и скачать необходимые вам файлы или же, наоборот, закачать свои файлы на сервер, если обладаете надлежащими правами доступа к серверу.

  В Интернете много публичных FTP-серверов, к которым разрешен анонимный доступ. Как правило, с таких серверов скачивать файлы разрешено всем желающим. Иногда, но очень редко, разрешается и запись файлов на публичный сервер. В состав любой операционной системы входит FTP-клиент — программа ftp. К тому же многие браузеры можно использовать в качестве FTP-клиента;
- SMTP (Simple Mail Transfer Protocol) — простой протокол передачи почты. Используется для отправки почты (e-mail);
- POP (Post Office Protocol) — протокол, используемый для получения сообщений электронной почты;
- IMAP (Internet Message Access Protocol) — еще один протокол для получения почты, но, в отличие от протокола POP, этот протокол позволяет читать почту без ее загрузки на компьютер пользователя.

  Протокол IMAP, по сути, намного удобнее, чем POP. Ведь почта хранится на сервере, и вы можете получить доступ к ней из любой точки земного шара, используя любой клиент. К тому же IMAP поддерживает поиск писем на сервере, что позволяет найти нужное письмо без загрузки всех писем на свой компьютер. Однако у IMAP есть один существенный недостаток, благодаря чему до сих пор распространен протокол POP, — IMAP требует постоянного соединения с сервером. Если нет соединения с сервером, вы не прочитаете не только новые сообщения, но и те, которые были получены ранее, поскольку все они хранятся на сервере. Так что в случае с IMAP об автономной работе (без подключения к Интернету) можно забыть.

22.6. Адресация компьютеров

Для идентификации узлов Интернета используются IP-адреса. IP-адрес представляет собой четыре числа, разделенные точками (можно сказать и иначе: одно 32-разрядное число, которое записывается в виде четырех восьмиразрядных чисел, разделенных точками). Нужно сразу отметить, что идентификация на основе IP-адресов не определяет компьютер в сети раз и навсегда однозначно, поскольку IP-адреса

бывают как присвоенными компьютеру постоянно (статические адреса), так и назначаемыми на текущий сеанс (динамические адреса). *Постоянные* (статические) IP-адреса обычно назначаются серверам, а *динамические* — обычным пользователям. Так что сегодня определенный динамический IP-адрес может быть назначен одному пользователю, а завтра — другому. Поэтому если в случае с аппаратными MAC-адресами еще можно говорить о какой-то однозначности (и то существуют способы подделки MAC-адресов), то IP-адреса по определению однозначными не являются.

Вот примеры IP-адресов: 127.0.0.1, 192.168.1.79, 111.33.12.99. Как было сказано ранее, IP-адрес — это одно 32-разрядное число или четыре 8-разрядных числа. Возведем 2 в восьмую степень и получим максимальное значение для каждого из четырех восьмиразрядных чисел — 256. Таким образом, учитывая, что некоторые IP-адреса зарезервированы для служебного использования, протокол IP (имеется в виду действующая сейчас четвертая версия протокола — IPv4) может адресовать примерно 4,3 млрд узлов. Однако с каждым годом количество узлов во Всемирной паутине увеличивается, поэтому была разработана шестая версия протокола IP — IPv6. Новый протокол использует 128-битные адреса (вместо 32-битных), что позволяет увеличить число узлов до 10^{12} и количество сетей до 10^{9}. IPv6-адреса отображаются как 8 групп шестнадцатеричных цифр, разделенных двоеточиями. Вот пример адреса нового поколения: 1628:0d48:12a3:19d7:1f35:5a61:17a0:765d.

> ***Примечание***
>
> Впрочем, массовый переход на IPv6 (который еще называют IPng — IP Next Generation) пока так и не состоялся, хотя его используют несколько сотен сетей по всему миру. В этой книге мы будем рассматривать только протокол IPv4, поскольку, судя по всему, Интернет не перейдет на IPv6 в ближайшие несколько лет. Интересующиеся могут прочитать об IPv6 по адресу: **http://ru.wikipedia.org/wiki/IPv6**.

IP-адреса выделяются *сетевым информационным центром* (NIC, Network Information Center). Чтобы получить набор IP-адресов для своей сети, вам надо обратиться в этот центр. Но, оказывается, это приходится делать далеко не всем. Существуют специальные IP-адреса, зарезервированные для использования в локальных сетях. Ни один узел глобальной сети (Интернета) не может обладать таким "локальным" адресом. Вот пример локального IP-адреса — 192.168.1.1. В своей локальной сети вы можете использовать любые локальные IP-адреса без согласования с кем бы то ни было. Когда же вы надумаете подключить свою локальную сеть к Интернету, вам понадобится всего один "реальный" IP-адрес — он будет использоваться на маршрутизаторе (шлюзе) доступа к Интернету.

Чтобы узлы локальной сети (которым назначены локальные IP-адреса) смогли "общаться" с узлами Интернета, используется специальная технология *трансляции сетевого адреса* (NAT, Network Address Translation). Маршрутизатор получает от локального узла пакет, адресованный интернет-узлу, и преобразует IP-адрес отправителя, заменяя его своим IP-адресом. При получении ответа от интернет-узла маршрутизатор выполняет обратное преобразование, поэтому нашему локальному узлу "кажется", что он общается непосредственно с интернет-узлом. Если бы мар-

шрутизатор отправил пакет как есть, то есть без преобразования, то его отверг бы любой маршрутизатор Интернета, и пакет так и не был бы доставлен к получателю.

Наверное, вам не терпится узнать, какие IP-адреса можно использовать без согласования с NIC? Об этом говорить пока рано — ведь мы еще ничего не знаем о *классах* сетей. IP-адреса служат не только для адресации отдельных компьютеров, но и целых сетей. Вот, например, IP-адрес сети — 192.168.1.0. Отличительная черта адреса сети — 0 в последнем октете.

Сети поделены на классы в зависимости от их размеров:

- класс A — огромные сети, которые могут содержать 16777216 адресов, IP-адреса сетей лежат в пределах 1.0.0.0 — 126.0.0.0;
- класс B — средние сети, содержат до 65536 адресов. Диапазон адресов — от 128.0.0.0 до 191.255.0.0;
- класс C — маленькие сети, каждая сеть содержит до 256 адресов.

Существует еще и классы D и E, но класс E не используется, а зарезервирован на будущее (хотя будущее — это IPv6), а класс D зарезервирован для служебного использования (широковещательных рассылок).

Представим ситуацию. Вы хотите стать интернет-провайдером. Тогда вам нужно обратиться в NIC для выделения диапазона IP-адресов на вашу сеть. Скажем, вы планируете сеть в 1000 адресов. Понятно, что сети класса C вам будет недостаточно. Поэтому можно или арендовать четыре сети класса C, или одну класса B. Но, с другой стороны, 65536 адресов для вас — много, и если выделить вам всю сеть класса B, то это приведет к нерациональному использованию адресов. Так что самое время ввести понятие *маски сети*. Маска сети определяет, сколько адресов будет занято сетью, фактически — маска задает размер сети. Маски полноразмерных сетей классов A, B и C представлены в табл. 22.1.

Таблица 22.1. *Маски сетей классов A, B и C*

Класс сети	Маска сети
A	255.0.0.0
B	255.255.0.0
C	255.255.255.0

Маска 255.255.255.0 вмещает 256 адресов (в последнем октете IP-адреса могут быть цифры от 0 до 255). Например, если адрес сети 192.168.1.0, а маска 255.255.255.0, то в сети могут быть IP-адреса от 192.168.1.0 до 192.168.1.255. Первый адрес (192.168.1.0) называется IP-адресом сети, последний — зарезервирован для широковещательных рассылок. Следовательно, для узлов сети остаются 254 адреса — от 192.168.1.1 до 192.168.1.254.

А вот пример маски сети на 32 адреса: 255.255.255.224 (255 – 224 = 31 + "нулевой" IP-адрес, итого 32).

Предположим, у нас есть IP-адрес произвольной сети, например, 192.168.1.0. Как узнать, к какому классу она принадлежит? Для этого нужно преобразовать первый октет адреса в двоичное представление. Число 192 в двоичной системе будет выглядеть так: **110**00000. Проанализируем первые биты первого октета. Если первые биты содержат двоичные цифры 110, то перед нами сеть класса C. Теперь проделаем то же самое с сетью 10.0.0.0. Первый октет равен 10, и в двоичной системе он будет выглядеть так: 00001010. Первый бит — 0, поэтому сеть относится к классу A. Опознать класс сети по первым битам первого октета поможет табл. 22.2.

Таблица 22.2. *Опознание класса сети*

Класс сети	Первые биты
A	0
B	10
C	110
D	1110
E	11110

Теперь поговорим о специальных зарезервированных адресах. Адрес 255.255.255.255 является *широковещательным*. Если пакет отправляется по этому адресу, то он будет доставлен всем компьютерам, находящимся с отправителем в одной сети. Можно уточнить сеть, компьютеры которой должны получить широковещательную рассылку, например, таким образом: 192.168.5.255. Этот адрес означает, что пакет получат все компьютеры сети 192.168.5.0.

Вам также следует знать адрес 127.0.0.1. Этот адрес зарезервирован для обозначения локального компьютера и называется *адресом обратной петли*. Если отправить пакет по этому адресу, то его получит ваш же компьютер, то есть получатель является отправителем, и наоборот. Данный адрес обычно используется для тестирования поддержки сети. Более того, к локальному компьютеру относится любой адрес из сети класса A с адресом 127.0.0.0. Поэтому при настройке реальной сети нельзя назначать компьютерам сети IP-адреса, начинающиеся со 127.

А теперь обратимся к IP-адресам сетей, зарезервированным для локального использования. В локальных сетях вы можете применять следующие адреса сетей:

- 192.168.0.0 — 192.168.255.0 — сети класса C (всего 256 сетей, маска 255.255.255.0);
- 172.16.0.0 — 172.31.0.0 — сети класса B (всего 16 сетей, маска 255.255.0.0);
- 10.0.0.0 — сеть класса A (одна сеть, маска 255.0.0.0).

Обычно в небольших домашних и офисных сетях используются IP-адреса из сети класса C, то есть из диапазона 192.168.0.0 — 192.168.255.0. Но поскольку назначение адресов контролируется только вами, вы можете назначить в своей локальной сети любые адреса, например, адреса из сети 10.0.0.0, даже если у вас в сети всего 5 компьютеров, и почувствовать себя администратором огромной сети.

22.7. Система DNS

Узлов в Интернете достаточно много, поэтому ни один человек не способен запомнить IP-адреса всех необходимых ему узлов — символьный адрес узлов **www.bhv.ru** или **www.dkws.org.ua** запомнить гораздо легче, чем их IP-адреса. Тем более, относительно недавно появилась возможность регистрации доменных имен на русском языке (точнее — на кириллице). Не знаю, приживутся ли такие доменные имена, но то, что они существуют, — это факт.

За преобразование IP-адресов в доменные имена и обратно отвечает *система доменных имен* (DNS, Domain Name System). Когда вы вводите доменное имя в строке браузера, система сначала разрешает (перетранслирует) это имя в IP-адрес (путем обращения к DNS-серверу), а потом подключается к узлу по полученному IP-адресу.

Впервые система DNS была представлена еще в 1984 году. Правда, тогда далеко не все сети перешли на использование DNS-серверов. В то время доменные имена разрешались в IP-адреса с помощью файла hosts, в котором содержалась таблица соответствия доменных имен IP-адресам. Понятно, что такой файл нужно было постоянно поддерживать в актуальном состоянии. Когда количество узлов увеличилось, и поддержка этого файла стала проблемой для администратора сети, вот тогда и началась эра DNS. Кстати, файлом hosts можно пользоваться до сих пор. Для обеспечения совместимости его поддержка включается даже в самые современные ОС (как в UNIX/Linux, так и в Windows), но, сами понимаете, обращаются к нему очень редко.

Система DNS более подробно будет рассмотрена в *главе 27*. Мы даже настроим собственный DNS-сервер.

22.8. Монтаж Ethernet-сети

22.8.1. Развитие стандарта Ethernet

Технология Ethernet намного "древнее", чем это можно себе представить. Хотя первая Ethernet-сеть была создана в 1975 году, она использует (как на момент создания, так и сейчас) метод доступа CSMA/CD *(см. разд. 22.8.2)*, который был разработан во второй половине 60-х годов прошлого века.

Создателем Ethernet-сети является компания Xerox. В 1980 году эта компания совместно с компаниями DEC и Intel разработала вторую версию стандарта Ethernet, которая называлась DIX (DEC, Intel, Xerox) или Ethernet-II. В то время Ethernet-сети использовали в качестве среды передачи данных только коаксиальный кабель.

Чуть позже был разработан стандарт IEEE 802.3, который практически полностью повторял стандарт Ethernet II. У нового стандарта были лишь небольшие отличия в формате кадров, в нем не было протокола тестирования конфигурации, который существовал в DIX, и новый стандарт выделял два уровня MAC и LLC, которые в DIX были одним целым.

Среди ранних модификаций стандарта Ethernet можно выделить следующие:

- Xerox Ethernet — оригинальный стандарт, предусматривающий скорость передачи данных 3 Мбит/с. Как уже было отмечено, существовали две версии этого стандарта: 1 и 2 (DIX);
- 10BROAD36 — позволял передавать данные на большие расстояния, для чего использовал технологию широкополосной модуляции. Средой передачи данных служил коаксиальный кабель. Стандарт не получил широкого распространения;
- 1BASE5 (второе название StarLAN) — дальний предок стандарта 10Base-T. Скорость передачи данных — 1 Мбит/с. В качестве среды передачи данных использовал витую пару. Не получил большого распространения.

Модификации стандарта Ethernet

После принятия стандарта IEEE 802.3 технология Ethernet продолжала бурно развиваться — скорость передачи данных повысилась до 10 Мбит/с (это еще в стандарте 802.3), а средой передачи данных служил уже не только коаксиальный кабель, но и витая пара, и оптоволоконный кабель. Модификации стандарта Ethernet представлены в табл. 22.3.

Таблица 22.3. *Модификации стандарта Ethernet*

Стандарт	Описание
10Base-5 (IEEE 802.3)	Использовался толстый коаксиальный кабель RG-8, именно поэтому данный стандарт иногда называют "толстый Ethernet". Максимальная длина сегмента — 500 метров
10Base-2 (IEEE 802.3a)	Сеть на базе тонкого коаксиала (кабель RG-58). Поскольку тонкий коаксиал более гибкий, такую сеть было проще монтировать, чем сеть 10Base-5. Однако максимальная длина сегмента не превышает 200 метров. Этот стандарт прожил долгую жизнь. Помню, видел сеть на его базе даже в 2002 году (правда, позднее уже не встречал таких, но уверен, что где-то они до сих пор есть)
StarLAN 10	Первый стандарт, работающий на скорости 10 Мбит/с и использующий витую пару. Послужил прототипом стандарта 10Base-T
10Base-T (IEEE 802.3i)	Среда передачи данных: витая пара 3-й или 5-й категорий. Максимальная длина сегмента такой сети — 100 метров
FOIRL (Fiber-optic inter-repeater link)	Стандарт, использующий для передачи данных оптоволоконный кабель. Послужил прототипом для 10Base-F. Максимальное расстояние передачи данных (без повторителя) — 1 км
10Base-F (IEEE 802.3j)	Основной стандарт, послуживший базой для стандартов 10Base-FL, 10Base-FB, 10Base-FP. Все стандарты используют в качестве среды передачи данных оптоволоконный кабель. Расстояние — до 2 км, скорость передачи данных — 10 Мбит/с

Таблица 22.3 *(окончание)*

Стандарт	Описание
10Base-FL (Fiber Link)	Практически то же самое, что и FOIRL, но длина передачи данных увеличена до 2 км
10Base-FB (Fiber Backbone)	Предназначался для объединения повторителей в магистраль
10Base-FP (Fiber Passive)	Топология "пассивная звезда", которая не предусматривает повторителей. Этот стандарт остался на бумаге — он не был реализован

Стандарты Fast Ethernet (100 Мбит/с)

Настоящий прорыв в развитии Ethernet произошел в 1995 году, когда появилась технология Fast Ethernet, позволяющая передавать данные со скоростью в 10 раз выше обычного Ethernet'а — 100 Мбит/с. С появлением Fast Ethernet коаксиальный кабель ушел в прошлое — новая технология в качестве среды передачи данных использовала только витую пару 5-й категории и оптоволоконный кабель.

Стандарты Fast Ethernet представлены в табл. 22.4.

Таблица 22.4. *Модификации стандарта Fast Ethernet*

Стандарт	Описание
100Base-T	Общее название стандарта для модификаций 100Base-TX, 100Base-T4 и 100Base-T, которые описаны далее. Все эти стандарты используют витую пару, а максимальная длина сегмента составляет 100 метров
100Base-TX (IEEE 802.3u)	Дальнейшее развитие стандарта 10Base-T. Как и в 10Base-T, используется топология "звезда"
100Base-T4	Создан из соображений обратной совместимости. Данный стандарт использует витую пару категории 3. Это значительно упрощает модернизацию сетей 10Base-T, где из соображений экономии применялась витая пара 3-й категории. Нужно отметить, что этот стандарт сейчас практически не используется
100Base-T2	Еще один вариант на витой паре 3-й категории, сейчас практически не применяется. Отличительная особенность: использует полный дуплекс (то есть один и тот же кабель может *одновременно* использоваться как для приема, так и для передачи данных). Скорость приема/передачи данных (в одном направлении) — 50 Мбит/с
100Base-FX	Использует многомодовое оптоволокно, максимальная длина сегмента в полудуплексе — 400 метров, в полном дуплексе — 2 км

Таблица 22.4 *(окончание)*

Стандарт	Описание
100Base-LX	Используется одномодовое волокно (оборудование на базе одномодового кабеля стоит дороже). Обеспечивает передачу данных на расстояние 15 км в режиме полного дуплекса, длина волны 1310 нм
100Base-LX WDM	То же, что и 100Base-LX, но допускается использование длин волны 1310 нм и 1550 нм. При этом с одной стороны используется передатчик с длиной волны 1310 нм, а с другой — 1550 нм

Что такое дуплекс?

В приведенной здесь таблице встретился, возможно, незнакомый вам термин — *дуплекс*. Существуют два режима передачи данных по одному и тому же кабелю: полудуплекс (Half Duplex) и полный дуплекс (Full Duplex). Рассмотрим оба режима на примере обычного телефона. Телефонная связь работает в *полнодуплексном* режиме, поскольку вы можете и говорить, и одновременно слышать своего собеседника, то есть вы можете говорить с ним одновременно. Если бы телефон работал в *полудуплексном* режиме, то когда бы вы говорили, то не слышали бы своего собеседника, поскольку передача шла бы в одном направлении — отправки. Вам нужно было бы сказать фразу, нажать какую-то кнопку, переключающую аппарат в режим приема информации, и тогда вы бы смогли услышать ответ своего собеседника.

Типы оптоволоконных кабелей

Многомодовый кабель — это кабель, где есть несколько пространственных мод, *одномодовый* — где имеется только одна мода. *Мода* — это тип электромагнитной волны в оптическом кабеле. Оптоволоконные кабели и сети на их основе из-за их дороговизны и сложности монтажа мы не рассматриваем. Интересующиеся могут прочитать о различных типах кабелей и их внутреннем устройстве по адресу: **http://kgg.moldline.net/teaching/cable/cable_media.htm**.

Gigabit Ethernet (1000 Мбит/с)

В 1998 году появилась новая технология — Gigabit Ethernet. Прорывом или революцией ее не назовешь. По сути, это количественное улучшение, а не качественное. Ничего нового создано не было: та же среда передачи данных, тот же метод разделения этой самой среды — CSMA/CD. Зато очень легко модернизировать сеть Fast Ethernet в Gigabit Ethernet — достаточно заменить сетевые адаптеры и коммутаторы. Кабели при этом останутся прежними, нужно будет только переобжать концевые коннекторы *(см. разд. 22.8.3)*. Модификации Gigabit Ethernet представлены в табл. 22.5.

Таблица 22.5. *Модификации стандарта Gigabit Ethernet*

Стандарт	Описание
1000Base-T (IEEE 802.3ab)	Использует витую пару категорий 5e или 6. В отличие от стандарта 100Base-TX, где используются только 2 пары кабеля (то есть 4 жилы), этот стандарт использует все 4 пары (8 жил), благодаря чему увеличивается скорость передачи данных

Таблица 22.5 *(окончание)*

Стандарт	Описание
1000Base-TX	Разработан Ассоциацией телекоммуникационной промышленности (Telecommunications Industry Association, TIA) в 2001 году. Работает в полном дуплексе, скорость передачи данных в обеих направлениях — 500 Мбит/с. Использует 2 пары (4 жилы) для передачи данных, и 2 — для приема, что упрощает конструкцию приемопередающих устройств, но требует витую пару более высокой категории — 6-й. Зато этот стандарт предполагает более простое оборудование, которое стоит дешевле, чем оборудование для 1000Base-T
1000Base-SX (IEEE 802.z)	Использует многомодовое оптоволокно, длина сегмента — 550 метров
1000Base-LX (IEEE 802.3z)	Дальность передачи данных при использовании многомодового оптоволокна — 550 м, а при использовании одномодового — до 40 км (без повторителей)
1000Base-CX	Подходит для передачи данных на небольшие расстояния (до 25 м) и использует экранированную витую пару (STP). Сейчас этот стандарт не применяется и заменен стандартом 1000Base-T
1000Base-LH (Long Haul)	Обеспечивает передачу данных на расстояние до 100 км без повторителей

Наше будущее — 10 Gigabit Ethernet

Относительно недавно был разработан новый стандарт, способный передавать данные со скоростью 10 Гбит/с, — 10 Gigabit Ethernet. Этот стандарт пока еще очень молод, и понадобится несколько лет, чтобы понять, какие его спецификации будут востребованы, а какие исчезнут с рынка.

Пока доступны следующие спецификации:

- 10GBase-CX4 — используется для передачи данных на короткие расстояния (до 15 м), применяются медный кабель CX4 и коннекторы InfiniBand. Мне кажется, что этот стандарт не получит особого распространения (как и 1000Base-CX), но поживем — увидим;
- 10GBase-SR — пригоден для передачи данных на небольшие расстояния (от 26 до 82 метров в зависимости от типа кабеля), использует многомодовое оптоволокно;
- 10GBase-LX4 — расстояние передачи данных от 240 до 300 метров по многомодовому оптоволокну или до 10 км по одномодовому оптоволокну;
- 10GBase-LR и 10GBase-ER — используются для передачи данных на расстояния до 10 и 40 км соответственно;
- 10GBase-T — принят в 2006 году (самый молодой стандарт из этого семейства), использует экранированную витую пару, длина сегмента (расстояние передачи данных) — 100 метров.

22.8.2. Несколько слов о коллизиях…

Чтобы иметь представление о Ethernet-сетях, вам нужно знать, что такое CSMA/CD (Carrier-Sense-Multiply-Access with Collision Detection) — метод коллективного доступа с обнаружением несущей (carrier) и коллизий. Этот метод используется во всех сетях с логической топологией "общая шина". Да, с появлением коммутаторов Ethernet-сети преобразились, но метод CSMA/CD служит до сих пор.

Представим себе общую шину — общую среду передачи информации. Ее можно сравнить с гирляндами лампочек на елке — все они подключены к одному проводу. Поскольку кабель общий, одновременно обмениваться информацией могут всего два компьютера. Спрашивается, какая же будет эффективность такой сети, если вся сеть должна ждать, пока два компьютера обмениваются информацией? Однако все происходит иначе. Все мы помним, что перед передачей данные разбиваются на части — пакеты. Общий алгоритм передачи данных таков:

1. Компьютер разбивает информацию на пакеты.
2. Затем он проверяет, не занята ли среда передачи данных.
3. Если среда свободна, компьютер передает один пакет.
4. После передачи пакета компьютер должен подождать 9,6 мкс, а потом начать процесс передачи следующего пакета.

Иногда случается *коллизия* — ситуация, когда два или больше компьютеров пытаются одновременно передать данные. Почему происходят коллизии, ведь компьютер перед отправкой данных проверяет, свободна ли среда передачи данных? Сначала разберемся, как он это делает. Компьютер прослушивает *несущую частоту* (carrier sense — вот откуда взялись две начальные буквы CS в названии метода!). Если несущей частоты (5–10 МГц) нет, то среда свободна. А теперь представим, что компьютер А только начал передавать кадр, а компьютер Б, который находится где-то очень далеко, одновременно начал проверять занятость среды передачи данных. Понятно, что несущая частота еще не "дошла" до компьютера Б, поэтому он тоже начал передачу данных. В результате передаваемые данные смешались — вот вам и коллизия…

Суть метода CSMA/CA в том и заключается, что когда два или большее количество узлов пытаются одновременно передать данные, CSMA/CA "просит" все узлы, кроме одного, прекратить передачу данных. "Счастливчик", которому разрешено передать данные, выбирается случайным образом. Но CSMA/CA может также предоставить приоритет узлу, который пытается передать данные, критические к времени (видео и/или голос).

В современных сетях на базе коммутаторов коллизии возникать, в общем-то, не должны — к каждому порту коммутатора подключено по одному компьютеру, и коммутатор передает пакет не всем компьютерам, а только тому, кому пакет адресован. Однако и в таких сетях коллизии порой возникают — например, когда сетевой адаптер и порт коммутатора одновременно начинают передавать кадры, решив, что кабель не занят. Правда, такая ситуация может сложиться только в полудуплексном режиме. В полнодуплексном режиме, как мы знаем, разрешена одновременная передача данных в обоих направлениях (прием и передача), поэтому в сети на базе коммутаторов, работающей в полнодуплексном режиме, коллизии не возникают.

22.8.3. Монтаж сети

Основные компоненты Ethernet-сети

Итак, давайте вспомним основные компоненты сети:

- *сетевые адаптеры* — с этим проблем сейчас нет, поскольку сетевые адаптеры встроены в материнские платы всех настольных компьютеров и ноутбуков;
- *вилки* (концевые коннекторы, разъемы) RJ-45 — ими обжимается витая пара, после чего обжатым кабелем можно соединить компьютер с коммутатором. Поскольку этими вилками кабель обжимается с обеих сторон, то количество вилок должно в два раза превышать количество компьютеров. Разъемы желательно покупать с запасом (они стоят копейки), поскольку во время обжима разъем легко повредить. На рис. 22.9 приведена схема нумерации контактов вилки RJ-45;
- *кабель "витая пара"* — о выборе витой пары мы поговорим чуть позже;
- *коммутатор* — перед покупкой коммутатора убедитесь, что он содержит количество портов, необходимое для подключения всех компьютеров вашей сети. Если у вас много компьютеров, скажем больше 24, то имеет смысл купить два коммутатора по 24 порта, чем один на 48 — для локализации трафика и уменьшения нагрузки на коммутатор (основные сетевые устройства рассмотрены в *приложении 4*);
- *инструмент для обжима витой пары* — именно этим инструментом вы будете обжимать витую пару (рис. 22.10).

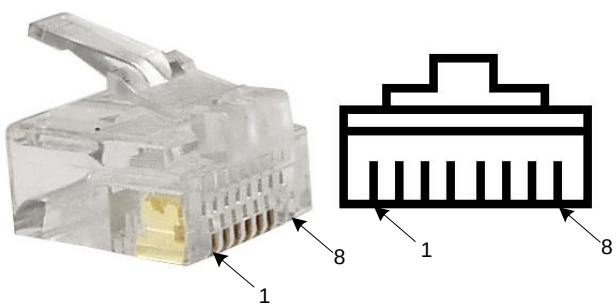


Рис. 22.9. Нумерация контактов вилки RJ-45



Рис. 22.10. Инструмент для обжима витой пары

Примечание

Интересно, что то, что практически все называют разъемом RJ-45, на самом деле называется вилкой 8P8C. Подробно об этом можно прочитать в Википедии: **http://ru.wikipedia.org/wiki/8P8C**.

Совет

Не покупайте дешевый инструмент для обжима витой пары! Помните пословицу о дешевой рыбке и соответствующей ей юшке? Когда-то я решил купить дешевый инструмент — он стоил в три раза дешевле, чем обычно. Да, на вид он был такой же.

> И вроде RG-45 обжимал. Как оказалось, именно "вроде". Сколько я разъемов испортил! Оказывается, этот инструмент не до конца обжимал разъем — обжимал не все его контакты. А чтобы обжать все контакты, приходилось сжимать разъем сильнее. Чуть-чуть перестарался, и все — разъем треснул. В итоге этот инструмент я выбросил и купил нормальный. Скупой платит дважды.

Если вам нужно построить небольшую сеть (скажем, до 10 компьютеров), тогда можно пойти путем наименьшего сопротивления и купить в компьютерном магазине уже готовые кабели. Они уже будут обжаты, и вам останется только соединить ими компьютеры с коммутатором. Обжатый кабель стоит немного дороже, чем комплект из витой пары необходимой длины и двух разъемов, но, учитывая, что компьютеров будет немного, а профессионально заниматься монтажом сети вы не собираетесь, то вы даже сэкономите. Хороший инструмент стоит дороже, чем 5–7 уже обжатых кабелей (в зависимости от длины). Правда, в случае с обжатыми кабелями есть один недостаток: они продаются только фиксированной длины — обычно по 5 или 10 метров, что не всегда удобно.

Подробнее о витой паре

Категории витой пары

Витая пара витой паре — рознь. Существуют различные категории витой пары. Тут все просто: чем выше категория, тем кабель более качественный и дорогой. Витая пара 1–4 категорий уже не используется для построения сетей (хотя для построения сетей и ранее использовалась витая пара только 3 и 4-й категорий, а кабели 1 и 2-й категорий применялись в телефонных сетях).

Для построения современных Ethernet-сетей используется витая пара 5 и 6-й категорий. Пятая категория (CAT5) представляет собой 4-парный кабель (4 пары жил) и служит для построения сетей 100Base-TX. В этих сетях задействованы всего 2 пары (4 провода), при этом достигается скорость передачи данных до 100 Мбит/с.

Более современная модификация пятой категории называется 5E. Сейчас практически вся продаваемая витая пара относится к этой категории. При покупке кабеля обратите внимание на надписи на его внешней оболочке — лучше приобрести витую пару именно CAT5E, чем просто CAT5, поскольку модернизированная версия может использоваться для построения сетей Gigabit Ethernet. Для передачи данных со скоростью 1000 Мбит/с используются все 4 пары.

В июне 2002 года появилась витая пара шестой категории — CAT6. Она стоит дороже, чем витая пара CAT5E. Если вы планируете использовать Gigabit Ethernet, то следует применить витую пару шестой категории — она лучше подходит для передачи данных со скоростью 1000 Мбит/с. К тому же, для скоростей 10 Гбит/с CAT5E не годится вовсе, нужна CAT6. Поэтому если вы в будущем собираетесь модернизировать свою сеть — ваш выбор CAT6.

Кроме 5 и 6-й есть еще и седьмая категория. Кабель седьмой категории имеет общий экран и экран вокруг каждой пары. По сути, кабель CAT7 является кабелем типа S/FTP (см. далее). Седьмая категория утверждена международным стандартом ISO11801 и поддерживает скорость передачи данных до 10 Гбит/с. Частота пропускаемого сигнала составляет 600–800 МГц.

Классификация витой пары по типу защиты

Существуют следующие типы витой пары:

- UTP (Unshielded twisted pair) — защита и экранирование отсутствуют, это самый дешевый вид витой пары и предназначен для использования внутри помещений (конечно, такой кабель можно использовать и снаружи, но в силу своей незащищенности долго он не прослужит);
- FTP (Foiled twisted pair) — есть один общий экран (для всех пар) из фольги. Тип более защищенный, чем UTP;
- STP (Shielded twisted pair) — экранированная витая пара, присутствует один экран для каждой пары;
- S/FTP (Screened Foiled twisted pair) — почти то же, что и FTP, но присутствует дополнительный внешний экран из медной оплетки;
- S/STP (Screened shielded twisted pair) — похож на STP, но присутствует дополнительный общий внешний экран.

Типы витой пары приведены по мере возрастания защиты и стоимости. Для наружного использования рекомендуется STP. Хотя если позволяют средства, то можно купить и S/STP.

Обжим витой пары

Обжать витую пару — это значит поместить отдельные жилы витой пары в определенной последовательности в вилку RJ-45 и закрепить их в этой последовательности с помощью инструмента.

А вот теперь начинается самое интересное: ключевая фраза здесь "в определенной последовательности". Последовательность расположения жил зависит от:

- стандарта кабеля;
- от того, кабель какого стандарта вы пытаетесь получить.

Существуют два стандарта витой пары: 568A и 568B (маркировка стандарта нанесена на оболочку кабеля). Если не вникать в подробности этих стандартов, то они для нас отличаются порядком обжима отдельных жил витой пары. И вся беда в том, что вам нужно помнить (или держать под рукой) обе схемы обжима. Ведь сегодня вы построили сеть на базе кабеля 568A, а завтра, когда пришлось подключить дополнительный компьютер, в продаже был только 568B. Мы рассмотрим обе схемы обжима, но чуть позже.

Схема обжима может зависеть не только от стандарта кабеля, но еще и от того, что вы хотите соединить. Ethernet-кабель бывает *прямым*, а бывает — *перекрестным* (его еще называют "кроссовер", от англ. cross-over).

- Прямой кабель используется для соединения компьютера с коммутатором и коммутатора с другим коммутатором, и схема обжима прямого кабеля с обеих сторон одинаковая.
- Перекрестный кабель служит для соединения двух компьютеров напрямую, без коммутатора (для организации сети из двух компьютеров) или для соединения некоторых старых коммутаторов/концентраторов, у которых есть порт Uplink. Одна вилка перекрестного кабеля обжимается как и у прямого кабеля, а вторая — перекрестно (с другим порядком расположения жил — чуть позже я поясню, как

именно). Поскольку у кроссовера порядок обжима витой пары изменен, его нельзя использовать для подключения компьютера к коммутатору.

Итак, когда мы знаем о стандартах 568A и 568B и о прямом и перекрестном обжиме, самое время приступить к обжиму. Напомню, что схема нумерации контактов вилки RJ-45 приведена на рис. 22.9.

Прямой кабель, Fast/Gigabit Ethernet

В табл. 22.6 приведена схема обжима прямого Ethernet-кабеля стандартов 568A и 568B для сети Fast/Gigabit Ethernet (100/1000 Мбит/с). Напомню, что прямой кабель обжимается одинаково с обеих сторон.

Таблица 22.6. *Прямой Ethernet-кабель (100/1000 Мбит/с)*

Номер контакта	Цвет жилы (для 568A)	Цвет жилы (для 568B)
1	Зелено-белый	Оранжево-белый
2	Зеленый	Оранжевый
3	Оранжево-белый	Зелено-белый
4	Синий	Синий
5	Сине-белый	Сине-белый
6	Оранжевый	Зеленый
7	Коричнево-белый	Коричнево-белый
8	Коричневый	Коричневый

Перекрестный кабель (кроссовер) для соединения 100 Мбит/с

В табл. 22.7 приводится номер контакта и схема обжима для первой стороны и для второй стороны кабеля. Обратите внимание: схема действительна только для кабеля 586B.

Таблица 22.7. *Кроссовер 100 Мбит/с*

Номер контакта	Сторона 1	Сторона 2
1	Оранжево-белый	Зелено-белый
2	Оранжевый	Зеленый
3	Зелено-белый	Оранжево-белый
4	Синий	Синий
5	Сине-белый	Сине-белый
6	Зеленый	Оранжевый
7	Коричнево-белый	Коричнево-белый
8	Коричневый	Коричневый

Перекрестный кабель (кроссовер) для соединения 1000 Мбит/с

В табл. 22.8 приводится схема обжима кабеля типа 586B по схеме кроссовера для соединения со скоростью 1000 Мбит/с (Gigabit Ethernet).

Таблица 22.8. *Кроссовер 1000 Мбит/с*

Номер контакта	Сторона 1	Сторона 2
1	Оранжево-белый	Зелено-белый
2	Оранжевый	Зеленый
3	Зелено-белый	Оранжево-белый
4	Синий	Коричнево-белый
5	Сине-белый	Коричневый
6	Зеленый	Оранжевый
7	Коричнево-белый	Синий
8	Коричневый	Сине-белый

Проверка правильности обжима кабеля

Обжимать кабель нужно тщательно, но стараясь не поломать коннекторы. Проверить, правильно ли вы обжали кабель, можно с помощью самого же коммутатора. Подключите один конец кабеля к компьютеру, а другой к коммутатору (коммутатор и компьютер должны быть включены). Напротив каждого порта коммутатора есть минимум два индикатора: первый (обычно маркируется "Link/ACT") показывает, есть или нет связь, а второй (может маркироваться "Speed", или "100", или "1000" — в зависимости от устройства) — скорость работы порта. Технология Fast Ethernet подразумевает передачу данных со скоростью 100 Мбит/с, но поддерживает и старые устройства, которые могут работать только на 10 Мбит/с. Так вот, второй индикатор загорается только в том случае, если обеспечивается скорость 100 Мбит/с.

Если же индикатор не загорается, давайте подумаем, в чем причина. Сетевой адаптер и коммутатор точно поддерживают скорость передачи данных 100 Мбит/с — старые устройства уже давно сняли с продажи, а новые помимо скорости 100 Мбит/с могут даже поддерживать скорость 1000 Мбит/с. Следовательно, дело в кабеле — вы неправильно его обжали (повреждение самого кабеля я исключаю), возможно, плохо обжался какой-нибудь из контактов. Попробуйте, не снимая вилку, еще раз обжать ее. Если опять не получится, нужно срезать вилку и обжать кабель заново.

В случае с Gigabit Ethernet все точно так же — если не загорается индикатор "Speed", то кабель обжат неправильно. Если оба индикатора не горят, нужно обжать кабель заново — и так до тех пор, пока не обожмете правильно.

22.8.4. Ограничения при построении сети

Правильно обжать вилки RJ-45 — это еще не все. Нужно помнить о минимальной и максимальной длине кабеля. Минимальная длина — 1 метр, меньше никак. Максимальная — 100 метров. Что делать, если 100 метров мало? В этом случае нужно использовать несколько коммутаторов: к одному коммутатору вы подключаете близлежащие компьютеры и второй коммутатор, а ко второму коммутатору — остальные компьютеры. На рис. 22.11 приведен пример построения такой сети, при котором расстояние между двумя максимально удаленными компьютерами получится 210 метров, что оптимально.

Хочу заметить, что максимальная длина сегмента 100 метров — это только по стандарту, на практике можно "выжать" значительно больше. В зависимости от сетевого адаптера и коммутатора возможна максимальная длина сегмента до 150 метров. Только сами понимаете, никто не гарантирует, что:

- такой сегмент вообще будет работать (в некоторых условиях будет работать, а в некоторых — нет);



Рис. 22.11. Схема простой, но "длинной" сети

- будет достигнута максимальная скорость. Скорее всего, максимум, что получится выжать из такого длинного сегмента, — 10 Мбит/с. При превышении максимального расстояния дальнейшее развитие событий зависит от коммутатора и сетевого адаптера. Как минимум, вы получите потери сигнала в 40% (следовательно, и потерю скорости).

Однако вы должны знать, что такой вариант возможен. Иногда расстояние от одного из компьютеров до коммутатора составляет 105–110 метров. Ради одного компьютера и десяти лишних метров покупать еще один коммутатор не хочется, поэтому можно попробовать работать с превышением максимальной длины. Может и получится, а может — нет...

Если нужно еще большее расстояние, то лучше использовать оптоволоконный кабель — в этом случае максимальное расстояние достигнет 2000 м. Но в этой книге мы не рассматриваем сети на базе оптоволоконного кабеля. Как правило, в домашних и офисных сетях среднего размера вполне можно обойтись витой парой.

В табл. 22.9 вы найдете ограничения для сетей Fast и Gigabit Ethernet.

Таблица 22.9. *Сводная таблица ограничений*

Характеристика	Fast Ethernet	Gigabit Ethernet
Минимальная длина кабеля	1 м	1 м
Максимальное расстояние между компьютером и коммутатором	100 м	100 м для 1000Base-T 25 м для 1000Base-CX
Максимальное расстояние между коммутаторами	90 м	90 м
Максимальный диаметр сети (расстояние между максимально удаленными друг от друга элементами)	210 (250) м	210 (250) м
Максимальное число компьютеров в сети	1024	1024
Витая пара (категория)	5, 5E, 6	5E, 6

А теперь предположим, что нам нужно построить сеть, подобную изображенной на рис. 22.11 (два сегмента и два коммутатора), но с максимально возможным расстоянием между коммутаторами? Из табл. 22.9 следует, что максимальное расстояние между коммутаторами может достигать 90 метров. Считаем: если максимальное расстояние от компьютера 1 до коммутатора 1 — 100 метров и от компьютера 2 до коммутатора 2 — тоже 100 метров, а максимальный диаметр сети равен 250 метрам (по данным табл. 22.9), то максимальное расстояние между коммутаторами может быть только 50 метров, а не 90, как указано в таблице. Обратите внимание, что и 250 метров — это теоретическое значение, на практике лучше все же ориентироваться на 210 метров (с запасом).

Иногда нужно увеличить расстояние между коммутаторами. Например, когда надо соединить два здания, находящихся на небольшом расстоянии друг от друга.

В каждом здании имеется коммутатор, к которому подключаются компьютеры, находящиеся в этом здании. Если длина кабеля между коммутаторами равна 90 метрам (теоретический предел), то максимальное расстояние от каждого коммутатора до конечного компьютера должно быть не более 80 метров: (250 – 90) / 2 (рис. 22.12). Да и это довольно-таки теоретические построения. Лучше, как было сказано, ориентироваться на максимальную длину сети 210 метров.

Что же касается максимального числа узлов сети (1024), то его с лихвой хватит для построения любой домашней и офисной сети среднего размера. Не забывайте также, что при большом количестве узлов можно использовать несколько 48-портовых коммутаторов, объединенных в стек.

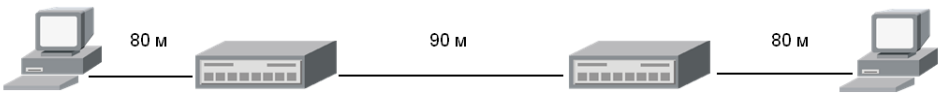


Рис. 22.12. Еще одна конфигурация сети

Глава 23



Настройка локальной сети

23.1. Определение имени сетевого интерфейса

В *главе 22* был рассмотрен монтаж Ethernet-сети, так что сейчас можно сразу приступить к настройке сети в FreeBSD. В случае наличия в вашей сети DHCP-сервера вся настройка сети сводится к добавлению в файл /etc/rc.conf одной строчки вида:

```
ifconfig_имя="DHCP"
```

Самое сложное — это указание имени. Честно говоря, мне после Linux, где сетевые платы назывались eth*N* (где *N* — число), было сложно привыкать к FreeBSD. Здесь имя сетевого Ethernet-интерфейса зависит от производителя адаптера. Так, имя em*N* используется для карт от Intel, а rl*N* — от Realtek. С одной стороны, взглянув на имя устройства, сразу понимаешь, каким чипом оно оснащено. С другой стороны, это было бы удобным, если бы в вашей системе стояло десять и более сетевых адаптеров. В Linux вы прописываете вызов нужного модуля ядра, обеспечивающего поддержку имеющейся сетевой карты, а далее используете имя eth0 (обычно в системе установлена одна сетевая карта, если не принимать во внимание компьютеры-шлюзы). В BSD же сначала нужно "вычислить" имя интерфейса, а потом приступать к настройке сети. В этом вам помогут данные табл. 23.1. Имя сетевого интерфейса будет состоять из имени устройства из табл. 23.1 и числа *N* — например, em0, rl0, fxp0 и т. д.

Таблица 23.1. *Имена сетевых адаптеров в FreeBSD*

Сетевая карта	Имя устройства
Broadcom BCM440x 10/100 Ethernet	bfe
Broadcom BCM570xx Gigabit Ethernet	bge
DEC/Intel 21143 и аналогичные платы на этом чипе	dc
DEC/Intel DC21x4x и аналогичные платы на этом чипе	de
Intel PRO/1000 Gigabit Ethernet (именно Gigabit Ethernet, а не обычный Ethernet)	em
Intel EtherExpress Pro/100B (82557, 82558)	fxp
Intel PRO/10GbE Ethernet Card	lxgb

Таблица 23.1 *(окончание)*

Сетевая карта	Имя устройства
Level 1 LXT1001 Gigabit ethernet	lge
NatSemi DP83820 Gigabit ethernet	nge
AMD Am79C79x PCI 10/100 NICs	pcn
RealTek 8139C+/8169/8169S/8110S	re
Классика: RealTek 8129/8139	rl
Adaptec AIC-6915 (Starfire)	sf
Silicon Integrated Systems SiS 900/SiS 7016	sis
SysKonnect SK-984x & SK-982x Gigabit Ethernet	sk
Sundance ST201 (D-Link DFE-550TX)	ste
Alteon Networks Tigon I/II Gigabit Ethernet	ti
Texas Instruments ThunderLAN	tl
SMC 9432TX	tx
3Com 3cR990	txp
VIA VT612x gigabit ethernet	vge
VIA Rhine, Rhine II	vr
3Com 3c590, 3c595	vx
Winbond W89C840F	wb
3Com 3c90x и платы на этом чипе	xl

Если вы не знаете, какой сетевой адаптер установлен в компьютере, а разбирать системный блок, чтобы узнать производителя и модель сетевой карты, не хочется, введите команду:

```
# dmesg | grep Ethernet
```

Вывод будет примерно таким:

em0: Ethernet address: XX:XX:XX:XX:XX:XX

Вы не только узнаете имя сетевого интерфейса (здесь **em0**), но и MAC-адрес интерфейса, что тоже пригодится.

Посмотреть более подробную информацию о сетевом интерфейсе можно, зная его имя:

```
# dmesg | grep em0
```

В моем случае вывод получился такой:

em0: <Intel(R) PRO/1000 Legacy Network Connection 1.0.1> port 0x2000-0x203f mem 0xd8920000-0xd893ffff,0xd8900000-0xd890ffff irq 19 at device 1.0 on pci2

em0: Memory Access and/or Bus Master bits were not set!

em0: [FILTER]

em0: Ethernet address: 00:0c:29:ed:05:47

Как можно видеть, сетевой адаптер поддерживает скорость 1000 Мбит/с (**PRO/1000**) и использует прерывание IRQ 19, указан также аппаратный порт и диапазон памяти. На практике эта информация пригодится при разрешении аппаратных конфликтов.

23.2. Настройка сетевого адаптера по DHCP

Итак, пока будем считать, что наш сетевой адаптер настраивается по DHCP. Добавьте в файл /etc/rc.conf строку:

```
ifconfig_em0="DHCP"
```

Сохраните файл и перезагрузите машину или введите команду `/etc/netstart`. Мой файл /etc/rc.conf на момент запуска `netstart` выглядел так:

```
hostname="denhost.localdomain"
ifconfig_em0="DHCP"
```

```
denhost# /etc/netstart
devd already running? (pid=555).
Setting hostuuid: 564d473f-0958-2a45-e2eb-2639dfed0547.
Setting hostid: 0xd8811a5c.
Starting Network: lo0 em0.
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
        options=3<RXCSUM,TXCSUM>
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
        inet6 ::1 prefixlen 128
        inet 127.0.0.1 netmask 0xff000000
        nd6 options=3<PERFORMNUD,ACCEPT_RTADV>
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM>
        ether 00:0c:29:ed:05:47
        inet 192.168.181.148 netmask 0xffffff00 broadcast 192.168.181.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
/etc/netstart: /etc/rc.d/ip6fw: not found
denhost#
```

Рис. 23.1. Перезапуск сети

Первый параметр (`hostname`) задает имя этого компьютера. Если он не задан, то имя установит DHCP-сервер или же будет использовано значение `localhost.localdomain`. Второй параметр — это настройка нашего сетевого интерфейса.

Вывод сценария `netstart` изображен на рис. 23.1: сетевому интерфейсу присвоен IP-адрес 192.168.181.148. Интерфейс lo0 — это интерфейс локальной петли (loopback), его IP-адрес 127.0.0.1 (данный интерфейс используется для тестирования поддержки сети и для обращения к локальному компьютеру).

После того как интерфейс будет настроен, можно узнать информацию о нем командой `ifconfig`. Эта команда используется не только для просмотра, но и для установки параметров интерфейса, но об этом — позже. Пока просто введите команду:

```
# ifconfig
```

em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500

options=9b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,
VLAN_HWCSUM>
ether 00:0c:XX:XX:XX:XX
inet 192.168.181.148 netmask 0xffffff00 broadcast 192.168.181.255
media: Ethernet autoselect (1000baseT <full-duplex>)
status: active
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
options=3<RXCSUM,TXCSUM>
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x3
inet6 ::1 prefixlen 128
inet 127.0.0.1 netmask 0xff000000
nd6 options=3<PERFORMNUD,ACCEPT_RTADV>

Из вывода команды `ifconfig` следует:

- у нашего сетевого адаптера MAC-адрес (аппаратный адрес) **00:0c:XX:XX:XX:XX** (параметр **ether**);
- адаптеру присвоен IP-адрес **192.168.181.148**;
- режим работы выбирается автоматически (**autoselect**), текущий режим **1000baseT <full-duplex>**;
- сетевой адаптер активен (подключен к сети), о чем свидетельствует его статус (**active**).

23.3. Настройка сетевого адаптера вручную

Теперь представим, что у нас нет DCHP-сервера и параметры сети нужно указать вручную. Проще всего продемонстрировать это на примере. Допустим, нам нужно присвоить IP-адрес 192.168.181.2, маску сети 255.255.255.0, имя узла denhost.localdomain и шлюз по умолчанию 192.168.181.1. В этом случае следует добавить в файл /etc/rc.config строки:

```
ifconfig_em0="inet 192.168.181.2  netmask 255.255.255.0"
defaultrouter="192.168.181.1"
hostname="denhost.localdomain"
```

23.4. Настройка сетевого адаптера с помощью конфигуратора *sysinstall*

Некоторые администраторы, особенно администраторы Windows, панически боятся (поначалу все так) редактирования конфигурационных файлов — они ведь привыкли к красивым графическим окошкам. В FreeBSD тоже есть конфигуратор, правда, псевдографический, но все равно это лучше, чем ничего.

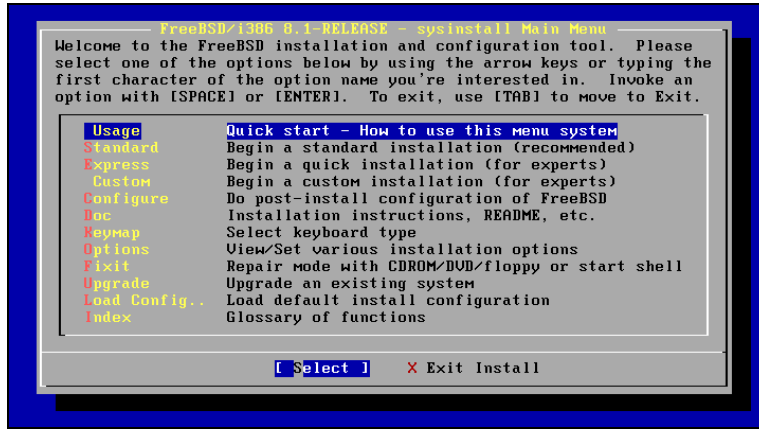


Рис. 23.2. Конфигуратор sysinstall

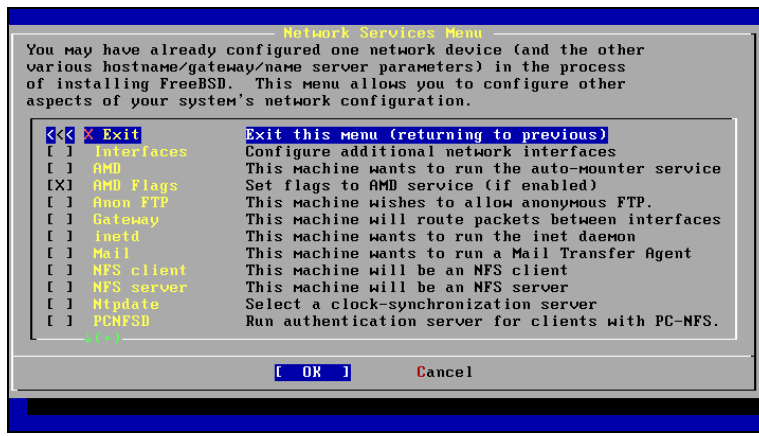


Рис. 23.3. Меню настройки сети

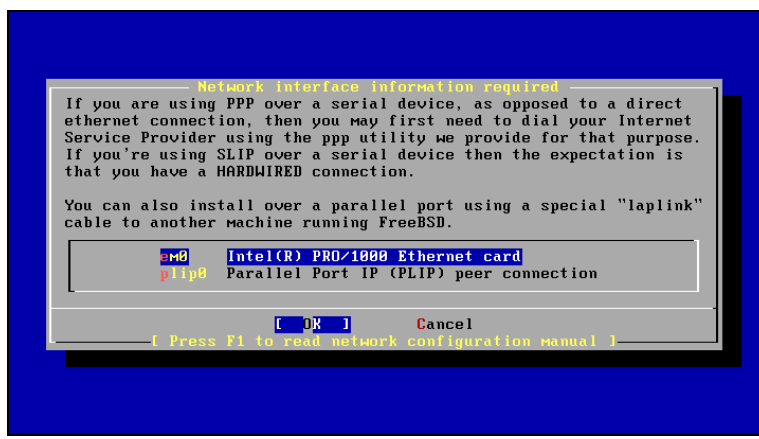


Рис. 23.4. Выбор сетевого интерфейса

Введите команду: `# sysinstall`

В открывшемся окне (рис. 23.2) выберите опцию **Configure | Networking**. В открывшемся меню настройки сети (рис. 23.3) выберите опцию **Interfaces**, а затем интерфейс, который вы хотите настроить (рис. 23.4).

Конфигуратор спросит вас, желаете ли вы использовать IPv6 для этого интерфейса и нужно ли настраивать этот интерфейс по DHCP. На первый вопрос целесообразно ответить **No**, поскольку IPv6 пока используется редко *(см. разд. 23.8)*, а ответ на второй вопрос зависит от наличия в вашей сети DHCP-сервера. Если DCHP-сервера нет, тогда нужно будет указать (рис. 23.5) имя узла (**Host**), домен (**Domain**), шлюз IPv4 по умолчанию (**IPv4 Gateway**), IP-адрес сервера имен (**Name server**), IP-адрес этого узла (**IPv4 Address**), маску сети (**Netmask**) и дополнительные параметры `ifconfig` (если нужно).

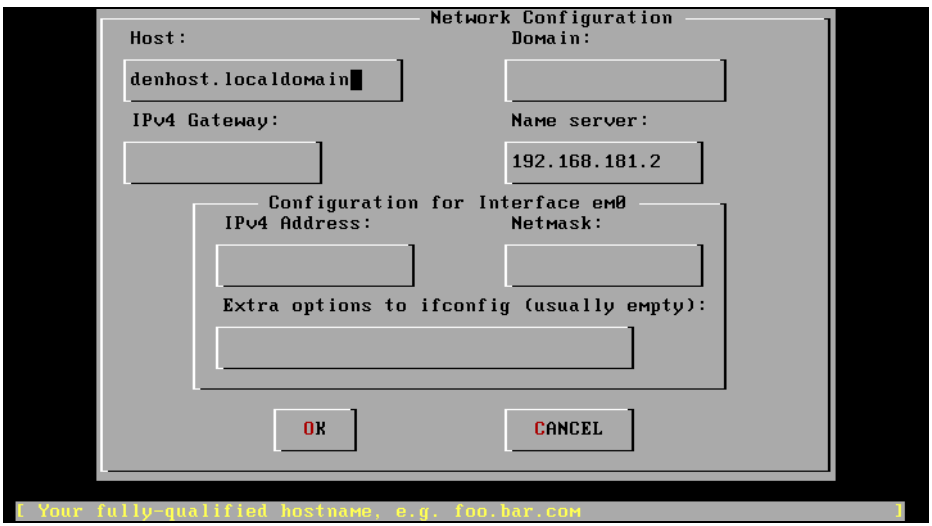


Рис. 23.5. Параметры сетевого адаптера

23.5. Настройка сетевого интерфейса с помощью команды *ifconfig*

Ранее было отмечено, что команду `ifconfig` можно использовать не только для получения информации, но и для настройки сетевого интерфейса. Рассмотрим несколько примеров использования `ifconfig` (для удобства я пронумеровал примеры):

```
1# ifconfig имя up
2# ifconfig имя down
3# ifconfig имя 192.168.181.5
4# ifconfig имя 192.168.181.5 netmask 255.255.255.0
5# ifconfig имя 192.168.181.5 netmask net0
```

```
6# ifconfig имя media 10baseT/UTP
7# ifconfig имя media 10baseT/UTP mediaopt full-duplex
8# ifconfig имя media autoselect
9# ifconfig имя lladdr <mac-address>
```

Первая команда "поднимает" сетевой интерфейс, вторая — "закрывает" сетевой интерфейс. Как правило, сначала нужно выполнить закрытие интерфейса. Потом — изменить его параметры, например, IP-адрес или MAC-адрес, а потом — поднять интерфейс.

Начнем с последней, 9-й команды — изменение MAC-адреса. Может пригодиться при тестировании сети. Например, когда вы настраиваете блокировку по MAC-адресу, можно определить, каким узлам следует предоставить доступ к DCHP-серверу, а каким нет. Сначала вы вносите MAC-адрес своего сетевого адаптера в "белый список" и проверяете, будет ли работать сеть. Затем, чтобы не переходить на другой компьютер, вы изменяете MAC-адрес и заново пытаетесь "поднять" сетевой интерфейс. Если все настроено правильно, DCHP-сервер не назначит вашему компьютеру IP-адрес, потому что его нет в "белом списке".

Впрочем, это только теория, возможно, есть и другие причины, по которым вам нужно сменить MAC-адрес. Например, мой провайдер устанавливает привязку к MAC-адресу. Это означает, что если у меня имя пользователя denis, то я не могу войти под этим пользователем с другого компьютера. А что делать, если мне нужно временно подключить другой компьютер и зайти в Интернет? Один из моих компьютеров используется в качестве медиацентра, и раз в несколько месяцев возникает потребность обновить программное обеспечение — мне намного проще подключить его к Интернету, обновить программное обеспечение, чем настраивать локальную домашнюю сеть и организовывать шлюз. Но подключиться я не могу, потому что имя пользователя привязано к MAC-адресу. Выход только один — изменение MAC-адреса, что можно выполнить командой `ifconfig`. Пример:

```
# ifconfig em0 down
# ifconfig em0 lladdr XX:XX:XX:XX:XX:XX
# ifconfig em0 up
```

Параметр `lladdr` — это псевдоним для параметра `ether`, поэтому в других руководствах по настройке сети вы можете встретить иную команду:

```
# ifconfig em0 ether XX:XX:XX:XX:XX:XX
```

Теперь перейдем к параметрам сети. Третья команда изменяет только IP-адрес интерфейса, четвертая — IP-адрес и маску сети:

```
# ifconfig имя 192.168.181.5
# ifconfig имя 192.168.181.5 netmask 255.255.255.0
```

Как изменить шлюз по умолчанию? Это делается командой `route`, которую мы рассмотрим в *разд. 23.6*.

Пятая команда тоже изменяет IP-адрес и маску сети, но вместо маски сети указывается идентификатор `net0`, прописанный в файле /etc/networks (формат этого файла прост и тщательно прокомментирован — просмотрите его на досуге):

```
# ifconfig имя 192.168.181.5 netmask net0
```

Три следующие команды — 6, 7 и 8-я — задают физические параметры интерфейса.

Примечание

Помню мое первое подключение по PPPoE к Интернету. Интернет работал довольно шустро (2 Мбит/с для 2005 года — это показатель!), но периодически происходили обрывы соединения. Конечно, было настроено переподключение *(см. разд. 24.4)*, но ситуация все равно изрядно напрягала. Коллега посоветовал "зажать" сетевой адаптер на скорости 10 Мбит/с (он у меня работал в режиме FastEthernet — 100 Мбит/с). На удивление — помогло. Разницы в скорости я не ощутил, поскольку скорость 10 Мбит/с в пять раз превышала лимит, установленный провайдером, зато никаких обрывов соединения (так и вспоминается ролик "Ни единого разрыва").

Итак, команда 6 задает полудуплексный режим работы на скорости 10 Мбит/с, команда 7 — полнодуплексный режим тоже на скорости 10 Мбит/с.

Пояснение

Полный дуплекс означает, что скорость 10 Мбит/с гарантируется в обоих направлениях: и при приеме данных и при их передаче. В режиме полудуплекса (half-duplex) скорость 10 Мбит/с — это суммарная скорость передачи данных (и на отправку, и на получение). Грубо говоря, чтобы добиться примерно такой же производительности, как в режиме полного дуплекса, скорость в режиме полудуплекса нужно удвоить.

Команда 8 задает автоматический способ определения режима работы сетевого адаптера (используется по умолчанию).

23.6. Команда *route*: маршрутизация

Команда `route` используется для управления таблицей маршрутизации. Управлять таблицей маршрутизации "по-настоящему" вам придется только при настройке шлюза/маршрутизатора, а этот процесс будет рассмотрен в *главе 28*. Пока лишь разберемся, как добавить шлюз (маршрут) по умолчанию:

```
# route delete default
# route add default IP-адрес
```

Первая команда удаляет маршрут по умолчанию на тот случай, если он был уже задан. Вторая — устанавливает новый шлюз (маршрут) по умолчанию. Суть маршрута по умолчанию, думаю, вам понятна. Когда компьютер А отправляет компьютеру Б данные, действия могут развиваться следующим образом:

- если компьютеры А и Б находятся в одной сети, компьютер Б просто получит данные;
- если компьютер Б находится в другой сети, система, используя таблицу маршрутизации, попытается отправить данные в сеть компьютера Б;
- если маршрута к сети, в которой находится компьютер Б, в таблице маршрутизации нет, тогда данные будут отправлены шлюзу по умолчанию;
- если шлюз по умолчанию недоступен или не установлен, будет получено сообщение **network unreachable** (сеть недоступна).

Просмотреть таблицу маршрутизации можно командой:

```
# netstat -rn
```

23.7. Имя узла, IP-адреса серверов DNS

Просмотреть и изменить имя узла можно командой `hostname`:

```
# hostname
```

localhost.localdomain

```
# hostname denhost.dkws.org.ua
# hostname
```

denhost.dkws.org.ua

Учтите, что команда `hostname` не вносит изменения в файл /etc/rc.conf, поэтому, чтобы имя узла сохранилось и после перезагрузки, следует изменить параметр `hostname` в файле /etc/rc.conf *(см. разд. 23.2)*.

Имя узла должно быть зарегистрировано на сервере DNS или хотя бы прописано в файле /etc/hosts. Формат этого файла прост:

```
IP-адрес имя имя.домен
```

Например:

```
127.0.0.1   localhost        localhost.localdomain
192.168.1.1 server           server.firma.ru
```

IP-адреса серверов DNS задаются в файле /etc/resolv.conf. Вот пример этого файла:

```
search firma.ru ru
namesever 192.168.181.1
namesever 192.168.181.2
```

Директива `search` пришла на смену устаревшей директиве `domain`, задающей имя домена, и используется для поиска домена. Например, когда пользователь хочет обратиться к узлу без указания домена (например, вводит команду `ping server`), к имени узла будет добавлено первое имя домена (firma.ru) и будет выполнено разрешение полученного доменного имени (server.firma.ru). Если системе не удалось разрешить первое имя, она аналогичным образом сформирует второе имя (server.ru) и попытается разрешить его. Домены, указывающиеся в директиве `search`, разделяются пробелами.

Директива `nameserver` задает IP-адрес сервера имен. Обычно есть два сервера имен: первичный и вторичный. Всего может быть четыре директивы `nameserver`.

Когда вы настраиваете DNS-сервер, то его файл /etc/resolv.conf должен выглядеть так:

```
search <ваш_домен>
nameserver 127.0.0.1
nameserver IP-адрес-сервера-DNS-провайдера
```

На остальных узлах файл /etc/resolv.conf должен выглядеть так:

```
search <ваш_домен>
nameserver IP-адрес-вашего-сервера-DNS-1
nameserver IP-адрес-вашего-сервера-DNS-2
```

При разрешении доменного имени в IP-адрес система сначала ищет IP-адрес узла в файле /etc/hosts, а потом уже обращается к DNS-серверу. Это поведение можно

изменить в файле /etc/nsswitch.conf. По умолчанию строка, задающая порядок разрешения доменных имен, выглядит так:

```
hosts: files dns
```

При желании можно сначала обращаться к DNS-серверу, а потом уже искать IP-адрес узла в /etc/hosts:

```
hosts: dns files
```

23.8. Несколько слов о поддержке IPv6

Может, эта книга попадет к вам в руки лет эдак через пять, и к тому времени переход на IPv6 будет уже выполнен. Хотя, когда точно будет выполнен этот переход, мне не известно — к сожалению, я не обладаю даром предсказания будущего. Но в любом случае, рано или поздно вам придется столкнуться с настройкой IPv6.

Для включения поддержки IPv6 в файл /etc/rc.conf нужно добавить строки:

```
ipv6_enable="YES"
ipv6_gateway_enable="YES"
```

Первая строка включает поддержку IPv6, а вторая — перенаправление пакетов IPv6 (IPv6 Forwarding).

Для настройки сетевого интерфейса используется модифицированная версия параметра `ifconfig`:

```
ipv6_ifconfig_имя="IPv6-адрес"
```

Например:

```
ipv6_ifconfig_em0="2001:db8:1033::3"
```

23.9. Суперсервер inetd

Сетевые сервисы (Web-сервер, почтовый сервер, FTP-сервер и др.) могут запускаться как автономно, так и через суперсервер inetd. У каждого способа есть свои преимущества и недостатки. При автономном запуске программа-сервер запускается при старте системы (чтобы обеспечить автоматический запуск в файл /etc/rc.conf добавляется соответствующая строка). После запуска программа поселяется в оперативной памяти, за ней закрепляется порт, и она ожидает запросов от пользователей по этому порту. Такая схема оправдывает себя, когда нагрузка на сетевой сервис большая, и он используется постоянно — ведь тогда при каждом запросе не придется запускать программу-сервер и не понадобится тратить время на ожидание ее запуска.

Однако, если сервис используется редко (например, вы развернули FTP-сервер для себя любимого и используете его раз в несколько дней), целесообразно запускать его через суперсервер inetd. При этом вместо, скажем, пяти программ-серверов в оперативную память будет загружен один суперсервер inetd. Когда поступит запрос на определенный порт (к примеру, на порт 21), суперсервер примет этот запрос, запустит программу-сервер (в данном случае FTP-сервер) и передаст ей полученный запрос.

Программы-серверы не будут запускаться при старте системы, что уменьшит время запуска системы, а также не будут забивать оперативную память, что положительным образом повлияет на производительность системы.

Основным файлом конфигурации суперсервера inetd является файл /etc/inetd.conf. Поскольку он довольно большой, я не стану его приводить полностью. Вместо этого мы рассмотрим строки, относящиеся к FTP-серверу (все строки однотипные, поэтому сказанное здесь будет актуально и для остальных сервисов). Вот заветные строчки:

```
ftp             stream tcp    nowait root   /usr/libexec/ftpd     ftpd -l
ftp             stream tcp6   nowait root   /usr/libexec/ftpd     ftpd -l
```

Вторая строка повторяет первую, но актуальна для протокола TCP6 — специальной версии TCP, которая используется в паре с IPv6. Допустимые протоколы указываются в файле /etc/protocols.

- Первое поле строки — это имя сервиса. По сути — имя протокола, оно должно совпадать с именем, указанным в файле /etc/services. В файле /etc/services нашему FTP-серверу соответствует запись:

```
ftp          21/tcp                     #File Transfer [Control]
ftp          21/udp                     #File Transfer [Control]
```

- Следующее поле — это тип сокета. Для TCP указывается `stream` (поток), для UDP — `dgram` (дейтаграмма).
- В третьей позиции указывается протокол. В нашем случае — протокол TCP (Transmission Control Protocol). Для других сервисов может быть указан протокол UDP (User Datagram Protocol, протокол пользовательских дейтаграмм).
- Четвертое поле может содержать значение `wait` или `nowait`. Первое значение обычно указывается для протокола UDP, второе — для TCP/TCP6.
- Пятое поле — имя пользователя, от имени которого будет запущена программа-сервис. Обычно сервисы запускаются от имени root, но некоторые сервисы из соображений безопасности запускаются от других пользователей со специальными правами. Например, сервис QMail запускается от имени пользователя qmaild, которому запрещен вход в систему, но разрешен доступ ко всем файлам и каталогам, которые могут понадобиться сервису QMail.
- Шестое поле — имя программы-сервера.
- Седьмое поле — аргументы, которые нужно передать программе-серверу.

Для автоматического запуска inetd нужно добавить в файл /etc/rc.conf строку:

```
inetd_enable="YES"
```

В Linux вместо суперсервера inetd давно используется его расширенная и усовершенствованная версия xinetd с совсем другим по формату конфигурационным файлом. Разработчики BSD более консервативны — здесь до сих пор работают с inetd. Впрочем, может, это и к лучшему — конфигурационный файл inetd.conf привычен многим, да и по структуре он проще, чем xinetd.conf.

23.10. Команды диагностики сети

Для диагностики сети вы будете использовать команды `ping`, `traceroute` и `netstat`. Утилита `ping` отправляет удаленному узлу ICMP-пакеты. Узел, получивший эти пакеты, отправляет нашему узлу ответ на "пинг". В результате можно узнать, доступен ли удаленный узел, и судить о качестве соединения — если запросов было отправлено больше, чем получено ответов, значит, часть запросов потерялась, следовательно, есть проблема с соединением).

Пример использования команды `ping`:

```
ping dkws.org.ua
```

```
PING dkws.org.ua (91.203.4.180): 56 data bytes
64 bytes from 91.203.4.180: icmp_seq=0 ttl=128 time=11.474 ms
64 bytes from 91.203.4.180: icmp_seq=1 ttl=128 time=12.067 ms
64 bytes from 91.203.4.180: icmp_seq=2 ttl=128 time=12.715 ms
64 bytes from 91.203.4.180: icmp_seq=3 ttl=128 time=12.420 ms

--- dkws.org.ua ping statistics ---
4 packets transmitted, 4 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 11.474/12.267/12.715/0.458 ms
```

Чтобы узнать, где именно теряются пакеты, нужно выполнить команду `traceroute` (в Linux для этого используется команда `tracepath`):

```
traceroute dkws.org.ua
```

Команда `traceroute` выведет список переходов (hop), то есть список маршрутизаторов, через которые проходят пакеты по пути к удаленному узлу. Если время перехода через какой-то маршрутизатор слишком большое, или же после какого-то маршрутизатора нет ответа (сообщение **no reply**), значит, нужно обратиться к администратору этого маршрутизатора и поинтересоваться, когда будет устранена проблема.

Команда `netstat` позволяет получить статистику "всея сети". Формат этой команды следующий:

```
netstat [-n] [-I интерфейс] интервал [система] [core]
netstat [-Aan] [-f семейство] [-I интерфейс] [-p протокол] [система] [core]
netstat [-n] [-s] [-i | -r] [-f семейство] [-I интерфейс] [-p протокол] [система] [core]
```

В зависимости от того, какую информацию вы хотите получить, и нужно использовать один из трех видов команды `netstat`.

Команда первого вида используется, когда нужно получить динамическую статистику пересылки пакетов по указанному интерфейсу с указанным интервалом. Интервал задается в секундах. Вот пример вывода команды `netstat 5` (обновление происходит каждые 5 секунд):

```
      input             (Total)           output
 packets  errs      bytes    packets  errs      bytes colls
       0     0          0          0     0          0     0
```

```
    0     0       0       0     0       0     0
    3     0     140       2     0     125     0
    0     0       0       0     0       0     0
    1     0      60       1     0      29     0
   11     0     795       5     0     187     0
    0     0       0       0     0       0     0
    0     0       0       0     0       0     0
   15     0     701       8     0     759     0
    6     0     260       2     0     208     0
```

Первые две колонки — это количество входящих пакетов и ошибок. Вторые две колонки — общее количество байтов, пакетов и ошибок. Далее приводится информация о входящих байтах и количестве коллизий. Команда `netstat` выполняется непрерывно, пока вы не нажмете комбинацию <Ctrl>+<C>.

Команда второго вида показывает открытые сокеты. При этом параметром `-I` вы можете задать сетевой интерфейс, с которого нужно снять статистику, а параметром `-p` установить фильтр протокола:

```
Active Internet connections
Proto Recv-Q Send-Q   Local Address          Foreign Address        (state)
tcp4       0      0 192.168.1.1.33624     host1.tuthost.c.ftp   ESTABLISHED
```

Из вывода ясно, что наша система (с адресом 192.168.1.1, локальный порт 33624) установила соединение с удаленным узлом host1.tuthost.c.ftp (это не полное имя узла, полное имя можно увидеть, если задать параметр `-n`, но тогда вывод не поместился бы на странице). Протокол FTP, протокол передачи — TCP4. Состояние соединения — установлено (**`ESTABLISHED`**). Могут быть и другие состояния соединения, указанные в табл. 23.2.

Таблица 23.2. *Состояния TCP-соединения*

Состояние	Описание
`CLOSED`	Соединение закрыто окончательно
`LISTEN`	Состояние прослушивания, сетевой сервис ожидает подключения
`SYN_SENT`	Попытка установить соединение
`SYN_RECEIVED`	Начальная синхронизация соединения
`ESTABLISHED`	Соединение уже установлено
`CLOSE_WAIT`	Ожидание закрытия соединения
`FIN_WAIT_1`	Сокет уже закрыт, ожидание отключения соединения
`CLOSING`	Процесс закрытия сокета: сначала отключилась наша сторона, а удаленная еще не успела
`LAST_ACK`	Наоборот: сначала отключилась удаленная сторона, а затем наша машина закрыла сокет

Таблица 23.2 *(окончание)*

Состояние	Описание
FIN_WAIT_2	Ожидание отключения удаленной системы
TIME_WAIT	Ожидание повторной передачи отключения удаленной стороны (уже после закрытия сокета)

Третья форма используется для выборки одной из сетевых структур данных — например, параметром `-r` можно показать таблицу маршрутизации:

```
# netstat -r
```

Параметр `-i` (обратите внимание — не `-I`) показывает статистику только автоматически сконфигурированных интерфейсов. Если вы настроили сетевой интерфейс вручную, он не будет показан.



Глава 24

Настройка DSL-соединения

DSL (Digital Subscriber Line) — цифровая абонентская линия, позволяющая производить двунаправленный обмен данными по телефонной линии. Существуют несколько вариантов DSL-линий: ADSL, VDSL, SDSL, RADSL. Наиболее распространены линии ADSL (Asymmetric DSL) — асимметричные цифровые линии. Для передачи данных используется витая пара телефонной сети. Скорость передачи зависит от расстояния до АТС — например, при расстоянии в 5–6 км можно получить 1,5 Мбит/с. Однако для абонентов скорость обычно ограничивается провайдером и зависит от тарифного плана. Самый доступный тарифный план подразумевает скорость передачи данных 64 Кбит/с.

24.1. Причина популярности DSL-соединений

Почему ADSL-соединения стали такими популярными? Основная причина популярности — это скорость и дешевизна. Именно эти два фактора. Как уже отмечалось, даже в самом дешевом варианте обеспечивается скорость передачи данных 64 Кбит/с — это в два раза быстрее, чем по обычному модему (конечно, в идеальных условиях из модема можно "выжать" 56 Кбит/с, но на практике это получается далеко не всегда). И при этом никаких разрывов соединений!

Да, за подключение к провайдеру нужно заплатить определенную сумму (напомню, что модемное подключение бесплатно), но, поверьте, оно того стоит. Также понадобится специальный ADSL-модем, который стоит дороже обычного модема, но в большинстве случаев есть возможность взять модем в аренду у провайдера, а стоимость такой аренды просто смешна.

Дешево, быстро — это все просто замечательно. Но имеется еще одно преимущество — когда вы работаете по ADSL в Интернете, ваш телефон не занят, в отличие от модемного соединения.

Впрочем, есть и нюанс — ADSL-соединение возможно только на цифровой телефонной линии.

24.2. Физическое подключение ADSL-модема

Схема подключения устройств показана на рис. 24.1. Специальное цифровое устройство (ADSL-сплиттер), обычно входящее в стандартный комплект поставки, подключено к телефонной линии. Простым телефонным кабелем к ADSL-сплиттеру

подключены обычный телефон и ADSL-модем. В свою очередь, ADSL-модем подключен к компьютеру с помощью отрезка Ethernet-кабеля (витой пары), также входящего в комплект поставки.

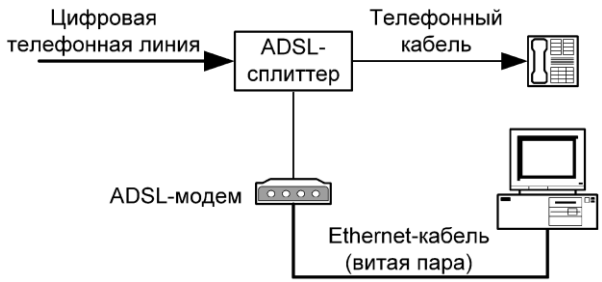


Рис. 24.1. Схема подключения ADSL-модема

Внимание!

Если у вас есть дополнительные параллельные телефоны, то подключать их к телефонной линии напрямую не допускается! Подключать параллельные телефоны можно только через ADSL-сплиттер.

24.3. Настройка соединения в FreeBSD

Настроить DSL-соединение в BSD достаточно просто, во всяком случае, проще, чем может показаться на первый взгляд. Первым делом нужно отредактировать файл /etc/ppp/ppp.conf (листинг 24.1).

Листинг 24.1. Файл /etc/ppp/ppp.conf

```
default:
       set device PPPoE:em0:isp
       set speed sync
       set mru 1492
       set mtu 1492
       enable lqr
       set ctsrts off
       set timeout 0
       set redial 0 0
       enable dns
isp:
       set authname имя
       set authkey пароль
       add default шлюз
```

При редактировании файла ppp.conf в большинстве случаев вам придется лишь заменить на свои значения, выделенные полужирным шрифтом. Во второй строке — это имя сетевого адаптера, к которому подключен модем. В трех последних строках нужно указать имя пользователя и пароль, а также IP-адрес шлюза по умолчанию (этот адрес можно уточнить в службе поддержки провайдера).

Впрочем, если нет желания общаться со службой поддержки провайдера, программа ppp может сделать это за вас. Вместо IP-адреса шлюза просто укажите значение `HISADDR`:

```
add default HISADDR
```

и адрес шлюза будет получен автоматически после подключения.

Если вы желаете использовать собственные DNS-серверы, указанные в файле /etc/resolv.conf, удалите строку `enable dns`, запрашивающую у PPPoE-сервера IP-адреса DNS-серверов провайдера.

> ***Примечание***
>
> Параметр `mtu` (Maximum Transmit Unit) задает максимальный размер пакета. По умолчанию данное значение может быть установлено автоматически, но не всегда оптимально. Если размер пакета окажется больше, чем позволяет машрутизатор провайдера, пакет будет разделен на несколько пакетов, что, естественно, скажется на скорости и пропускной способности соединения. Если размер пакета получится меньше, чем положено, тоже не хорошо — канал будет использован нерационально, ведь станут проходить полупустые кадры. Поскольку мы работаем по протоколу PPPoE, нужно учитывать несколько факторов. Максимальный размер кадра Ethernet составляет 1518 байтов, из которых 18 уходит на заголовок и контроль, поэтому для полезных данных остается 1500 байтов. Обычно данное значение и указывается для Ethernet. Но ведь по Ethernet мы собираемся передавать пакеты PPP, и PPPoE отбирает еще 6 байтов, а PPP — 2 байта. Таким образом получается, что для PPPoE значение `mtu` должно быть равно 1492. Параметр `mru` (Maximum Receive Unit) обычно равен параметру `mtu`.

Если вам нужно предоставить данное соединение в общее пользование (чтобы другие компьютеры локальной сети могли подключиться к Интернету через ваше соединение), после параметров секции `default` добавьте следующие строки:

```
nat enable yes
nat log yes
nat same_ports yes
nat unregistered_only yes
enable dns
```

Так будет включена NAT (Network Address Translation, трансляция сетевых адресов), но не нужно думать, что этим мы настроили полноценный шлюз (о настройке шлюза разговор пойдет в *главе 28*).

Сохраните файл и введите команду установки соединения:

```
# ppp -ddial isp
```

После этого попробуйте пропинговать какой-нибудь удаленный узел, например:

```
# ping dkws.net
```

Если ответа не будет, и возникнет подозрение, что соединение не установлено, просмотрите файлы /var/log/messages, /var/log/ppp.log и вывод команды `ifconfig` (отобразит список активных интерфейсов).

Осталось только настроить автоматическое подключение к провайдеру при запуске системы. Добавьте в файл /etc/rc.conf следующие строки:

```
ppp_enable="YES"
ppp_mode="ddial"
ppp_profile="isp"     # имя профиля провайдера из ppp.conf
ppp_nat="YES"         # если нужен NAT, иначе укажите NO
```

24.4. Управление переподключением

Давно прошли времена вечных обрывов соединения (да, телефонные модемы мы будем помнить долго), но DSL-соединение тоже может обрываться. Иногда это происходит чаще, чем мы ожидаем. Автоматическое переподключение можно организовать путем отправки запроса проверки качества (Link Quality Request). Для этого в секцию `default` файла ppp.conf добавьте строки:

```
enable lqr echo
set lqrperiod 60
```

Первая строка включает запрос проверки качества, а вторая устанавливает периодичность его отправки в секундах. Каждые 60 секунд система станет отправлять запрос проверки качества соединения. Если сервер провайдера не ответит, будет выполнено автоматическое переподключение.

Можно использовать и старые методы, работающие даже в самых ранних версиях FreeBSD: параметры `redial` и `reconnect`:

```
set reconnect 3 5
set redial 5 10
```

Первая команда указывает демону pppd восстанавливать подключение в случае потери несущей. Количество попыток — пять с паузой в три секунды между попытками. Если указать значения `0 0`, автоматическое переподключение будет выключено.

Вторая команда относится к подключению. Иногда при подключении к серверу провайдера может возникнуть ошибка — например, сервер не отвечает по причине его перезагрузки (ошибка 718 в Windows), или же зависла сессия пользователя (обычно это приводит к ошибке 691 в Windows). В этом случае нужно "перезвонить" провайдеру еще раз (терминология уходит своими корнями в эру коммутируемых модемных соединений, несмотря на то, что уже давно используется DSL/PPPoE). Первое значение задает паузу между попытками, а второе — количество попыток. Слишком много попыток указывать не нужно, причина может быть самой банальной — неоплата доступа в Интернет.

Глава 25



Подключение к сети Windows

25.1. Установка Samba

BSD — отличная операционная система, но от Windows нам не уйти. Windows будет окружать нас всегда, будь то домашняя, корпоративная сеть или интернет-кафе. Нам постоянно предстоит обмениваться документами с Windows-компьютерами — ведь далеко не все пользователи предпочитают работать в BSD или Linux.

В BSD для взаимодействия с сетью Microsoft служит порт samba34 (/usr/ports/net/samba34). Если вы хотите использовать общие ресурсы сети Windows, установите этот порт. Он позволяет не только пользоваться общими ресурсами сети, но и предоставлять собственные ресурсы Windows-пользователям. Причем все происходит так, что Windows-пользователи даже не заметят разницы.

Название порта зависит от версии Samba:

- /usr/ports/net/samba34 — Samba 3.4;
- /usr/ports/net/samba33 — Samba 3.3;
- /usr/ports/net/samba32 — Samba 3.2;
- /usr/ports/net/samba3 — Samba 3.0.

Кроме порта samba3* вам также понадобится (для обращения к ресурсам Windows-сети) порт samba-client, содержащий клиентскую часть Samba.

Приступим к установке Samba из портов:

```
# cd /usr/ports/net/samba34
# make install
```

Можно также установить Samba с помощью команды `pkg_add`:

```
# pkg_add -r samba34
```

> ***Примечание***
>
> Учтите, что пакет samba3*, который устанавливается с помощью команды `pkg_add`, не содержит поддержки ADS (Active Directory Service). Чтобы добавить эту функциональность (опция так и называется — ADS), нужно собирать Samba из портов. При установке из портов также можно включить опцию SWAT — Web-интерфейс управления Samba, позволяющий быстро настроить общие ресурсы.

Добавьте в файл rc.conf строку запуска Samba:

```
samba_enable="YES"
```

Затем запустите Samba:

```
# /usr/local/etc/rc.d/samba start
```

25.2. Файл конфигурации Samba

Основным файлом конфигурации Samba является файл /usr/local/etc/smb.conf. Откроем его и изменим несколько параметров. Первым делом измените параметр `WORKGROUP` — он задает имя рабочей группы или домена NT:

```
WORKGROUP = MSHOME
```

Конечно, имя группы у вас, скорее всего, будет другим. Можете также установить параметр `server string` — это описание вашего компьютера:

```
server string = My FreeBSD computer
```

> ***Пояснение***
>
> В более ранних версиях Samba параметр `server string` назывался `comment`.

Установите параметр `security`. Если у вас сеть клиент/сервер, следует выбрать значение `server`, а если сеть одноранговая (то есть сеть без выделенного сервера), то выберите параметр `user` или `share`:

```
security = share
```

Если вы выбрали параметр `user`, надо будет добавить Samba-пользователей для доступа к ресурсам этого компьютера.

Сначала добавим пользователя `samba1`, указав оболочку /sbin/nologin и отключив тем самым обычный вход в систему:

```
# pw useradd samba1 -c "Samba User" -s /sbin/nologin
```

Затем изменим обычный и Samba-пароли для этого пользователя:

```
# passwd samba1
# smbpasswd samba1
```

Имя гостевой учетной записи установите так:

```
guest account = guest
```

Нужно настроить и кодировки:

```
unix charset = UTF-8
dos charset = UTF-8
display charset = UTF-8
```

> ***Пояснение***
>
> В более ранних версиях Samba для настройки кодировок использовались параметры `client code page` и `character set`. Теперь они не поддерживаются и заменены параметрами `unix charset` (задает кодировку, в которой хранятся файлы конфигурации Samba), `dos charset` (задает кодировку для Windows-клиентов) и `display charset` (задает кодировку для Samba-клиентов).
>
> Текущая версия Samba полностью поддерживает кодировку UTF-8 (как и современные версии Linux и Windows), поэтому проблем с UTF-8 возникнуть не должно.

Параметр `interfaces` указывает интерфейсы, на которых должен работать сервис `smb`. Укажите те интерфейсы, которые связывают вашу машину с Windows-сетями:

```
interfaces = 192.168.0.22/24
```

Для того чтобы Samba работала быстрее, установите следующие опции (а что они означают, мы рассмотрим в *разд. 25.4*):

```
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
dns proxy = no
```

25.3. Настройка общих ресурсов

Теперь осталось сконфигурировать ресурсы, которые вы хотите предоставить в общее пользование (листинг 25.1). Фрагмент, приведенный в листинге 25.1, нужно добавить в файл конфигурации Samba /usr/local/etc/smb.conf.

Листинг 25.1. Секция `[public]`

```
[public]
# общий каталог, комментарий для ресурса задается директивой comment
comment = Public Directory
# путь
path = /var/samba
# не только чтение
read only = no
# разрешить запись
writable = yes
# разрешить гостевой доступ
guest ok = yes
# разрешить просмотр содержимого каталога
browseable = yes
```

> ***Примечание***
>
> Ранее директива `comment` задавала и описание сервера, и описание разделяемого ресурса. Сейчас директива `comment` для сервера заменена директивой `server string`, а ресурсы по-прежнему описываются директивой `comment`.

Здесь общим ресурсом нашего компьютера объявлен каталог /var/samba. В него другие пользователи смогут записывать свои файлы (`read only = no`, `writeable = yes`), естественно, они смогут их и читать (`browseable = yes`). Проверка имени пользователя и пароля для доступа к ресурсу не нужна (`guest ok = yes`) — используется так называемый *гостевой* доступ. Комментарий `Public Directory` увидят другие пользователи Windows-сети при просмотре ресурсов вашего компьютера.

Рассмотрим еще один пример, позволяющий сделать общими домашние каталоги пользователей (листинг 25.2).

Листинг 25.2. Секция `[homes]`

```
[homes]
 comment = Home Directories
 browseable = no
 valid users = %S

# запись запрещена, только просмотр
 writable = no

# маска при создании файлов, нужна, если writable = yes
create mask = 0600
# маска при создании каталогов, нужна, если writable = yes
directory mask = 0700
```

В листинге 25.3 приведен пример предоставления общего доступа к CD/DVD. Будем считать, что наш CD/DVD смонтирован в каталог /cdrom.

Листинг 25.3. Пример общего доступа к CD/DVD

```
[cdrom]
   comment = Samba server's CD-ROM
   writable = no
   locking = no
# каталог /cdrom должен существовать и являться
# точкой монтирования CD/DVD
   path = /cdrom
   public = yes
   preexec = /bin/mount /cdrom
   postexec = /bin/umount /cdrom
```

Теперь разберемся, как предоставить в общее распоряжение принтеры. В *главе 19* уже отмечалось, что для организации печати нужно установить систему печати CUPS (порт cups), а для того, чтобы Samba могла предоставлять в общий доступ принтеры, которыми управляет CUPS, нужен еще и порт cups-samba. Процесс установки портов cups и cups-samba подробно описан в *разд. 19.4*.

После установки портов cups и cups-samba в файл конфигурации Samba (smb.conf) следует добавить (читайте — просто раскомментировать) строки:

```
load printers = yes
printing = cups
printcap name = cups
```

Первая строка обеспечивает загрузку принтеров, а две следующие указывают, что нужно использовать систему печати CUPS. Затем необходимо раскомментировать следующие строки, обеспечивающие экспорт принтеров:

```
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
public = yes
guest ok = yes
writable = no
printable = yes
printer admin = root

[print$]
comment = Printer Drivers
path = /usr/local/share/cups/drivers
browseable = yes
# guest ok = yes тоже работает
guest ok = no
read only = yes
write list = root
```

Теперь перезапустите Samba:

```
# /usr/local/etc/rc.d/samba restart
```

Принтеры добавляются в общее использование командой:

```
# cupsaddsmb -U root имя
```

Например:

```
# cupsaddsmb -U root lexmark
```

Если вам повезет, то принтер будет добавлен и сразу станет доступным для других узлов. А вот если нет, придется совершить ритуал танца с бубном... Поскольку я не знаю, что именно у вас не получится, советы могут быть только общими. Одна из возможных проблем описывается (и решается) в следующей статье: **http://www.opennet.ru/base/net/freebsd_cups_samba_ad.txt.html**.

25.4. Оптимизация Samba

В файле конфигурации smb.conf имеется параметр `wide links`. Никогда не устанавливайте его в `no`! — так вы существенно снизите производительность Samba. Наоборот, если вы установите его в `yes` (если до этого параметр `wide links` был отключен), производительность Samba существенно повысится.

Дело в том, что параметр `wide links` определяет, как Samba будет следовать по символическим ссылкам. Если `wide links = no`, Samba не будет следовать по символическим ссылкам вне экспортируемой области. При этом сначала Samba следует

по символической ссылке, а затем выполняет так называемый *directory path lookup* (системный вызов, определяющий, где завершилась ссылка). Эта операция подразумевает на шесть системных вызовов больше, нежели в случае, если `wide links = yes`. Учитывая, что подобных операций делается очень много, выключение `wide links` снижает производительность Samba приблизительно на 30%.

Вернемся к параметру `dns proxy`. Если этот параметр установлен в `yes` (кстати, это и есть значение по умолчанию), то демон nmbd переправляет запросы на определение имени серверу DNS (если имя не найдено в базе данных WINS). Включать параметр `dns proxy` имеет смысл только, если Samba-сервер используется как сервер WINS. В противном случае включение данного параметра снизит производительность работы Samba без получения каких-либо преимуществ. Поэтому я и рекомендовал выключить этот параметр:

```
dns proxy = no
```

Протокол TCP/IP — штука тонкая. Производительность сетевых приложений во многом зависит от того, правильно ли настроен TCP/IP. Samba — настоящее сетевое приложение, которое к тому же работает по протоколу TCP/IP. При использовании TCP/IP, если размер запросов и ответов не фиксирован (как в случае с Samba), рекомендуется применять протокол TCP с опцией `TCP_NODELAY`. Для этого в файл smb.conf нужно добавить строку:

```
socket options = TCP_NODELAY
```

Тесты показывают, что с указанными опциями Samba при больших нагрузках работает в три раза быстрее, чем без них. Если Samba используется в локальной сети (в большинстве случаев так оно и есть) рекомендуется еще указать и такую опцию `IPTOS_LOWDELAY`:

```
socket options = IPTOS_LOWDELAY TCP_NODELAY
```

Если есть желание "выжать" из Samba еще больше, тогда установите следующие параметры буферизации: `SO_RCVBUF=8192 SO_SNDBUF=8192`. Например:

```
socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
```

25.5. Программа smbclient

Программа smbclient позволяет использовать общие ресурсы других компьютеров в Windows-сети. В простейшем случае программа smbclient вызывается так:

```
$ smbclient \\сервер\ресурс [пароль]
$ smbclient \\сервер\ресурс [пароль] -U имя_пользователя
```

Пароль можно не указывать, если сервер (компьютер, предоставляющий ресурс) его не требует. Далее работа с smbclient напоминает работу с FTP-клиентом ftp. Вы можете использовать команду `put` для записи файлов в общий ресурс (если у вас есть права доступа), `get` для получения файла с общего ресурса, `cd` — для смены каталога, `dir` — для вывода объявления каталога и т. д. Если вы умеете работать с программой ftp, то без особых проблем справитесь и с программой smbclient.

Глава 26



DHCP-сервер

26.1. Протокол динамической конфигурации узла

DHCP (Dynamic Host Configuration Protocol, протокол динамической конфигурации узла) используется для автоматической настройки узлов сети. С помощью DHCP компьютер, подключенный к сети, в которой есть DHCP-сервер, может получить IP-адрес, маску сети, IP-адрес шлюза, адреса серверов DNS и другие сетевые параметры.

Особенно удобно использовать DHCP в средних и больших сетях. Вы только представьте, что ваша сеть включает, скажем, 20 компьютеров. Если компьютерам сети назначать IP-адреса статически, то вам придется подойти к каждому компьютеру и указать его IP-адрес. Заодно потребуется ввести IP-адрес сети, IP-адрес шлюза и адреса серверов DNS. Понятно, что эту процедуру надо будет выполнить разово — при настройке сети. Но если через некоторое время конфигурация сети изменится (например, вы поменяете провайдера, и IP-адреса DNS-серверов изменятся), вам придется все повторить заново — подойти к каждому компьютеру и прописать изменившиеся адреса.

Если же потратить полчаса на настройку DHCP-сервера, можно будет централизованно управлять конфигурацией сети. Тогда стоит только изменить IP-адрес DNS-сервера в конфигурационном файле DHCP-сервера — на остальных компьютерах сети новые IP-адреса DNS-серверов пропишутся автоматически. Удобно? Я тоже так думаю.

Рассмотрим, как работает DHCP-сервер. После подключения к сети клиент, настроенный на использование DHCP, производит по протоколу UDP широковещательную рассылку (адрес 255.255.255.255, порт 68) с целью обнаружения DHCP-сервера. Сервер, получив такой запрос от клиента, назначает ему один из свободных IP-адресов (передача параметров осуществляется на порт 67). Кроме адреса сервер передает клиенту и другие сетевые параметры, заданные администратором (IP-адреса DNS-серверов, IP-адрес шлюза, маску сети и т. д.). IP-адрес выделяется не навсегда, а на определенное время — на время *аренды* (задается в конфигурационном файле сервера). По истечении времени аренды IP-адрес будет возвращен в список свободных адресов, а клиенту будет назначен новый адрес. Иногда сервер

назначает клиенту адрес, который он использовал в прошлый раз, но 100%-й гарантии быть не может.

DHCP-сервер можно настроить так, что определенному клиенту он будет назначать фиксированный адрес. В этом случае сервер проверяет MAC-адрес сетевого адаптера клиента и, если он совпадает с указанным в конфигурационном файле, сервер назначает этому клиенту его фиксированный IP-адрес.

Для установки DHCP-сервера необходимо установить порт net/isc-dhcp3-server (установка программного обеспечения рассмотрена в *главе 18*). Если вам нужен не только DHCP-сервер, но и DHCP-клиент (например, на другой BSD-машине), тогда установите пакет dhcpclient.

В отличие от Linux, где сетевые интерфейсы автоматически настраиваются на использование DHCP, FreeBSD придется настраивать вручную. Откройте файл /etc/rc.conf, найдите строку параметров сетевого интерфейса (начинается с `ifconfig_`), присвойте ей значение DHCP. Например:

```
ifconfig_em0="DHCP"
```

Для перезапуска сети введите команду:

```
# /etc/netstart
```

Можете по примеру Microsoft перезагрузить весь компьютер (командой `reboot`).

26.2. Конфигурационный файл DHCP-сервера

Конфигурационным файлом DHCP-сервера является файл /usr/local/etc/dhcpd.conf. Пример его содержимого вы можете найти в файле /usr/local/etc/dhcp.conf.sample.

> ***Внимание!***
>
> Директивы конфигурационного файла /usr/local/etc/dhcpd.conf не чувствительны к регистру символов, то есть вы можете написать как `option`, так и `OPTION`, но принято писать строчными буквами. Комментарии в этом файле начинаются с символа решетки `#`.

В начало файла конфигурации нужно поместить одну из директив:

```
ddns-update-style ad-hoc;
```

или

```
ddns-update-style interim;
```

> ***Примечание***
>
> Разберемся, о чем речь. Сейчас существуют две схемы обновления DNS: непосредственное обновление (ah-doc) и предварительное взаимодействие DHCP–DNS (interim). Вторая схема пока не утверждена комитетом по техническому развитию Интернета, но уже успешно используется, и разработчики DHCP рекомендуют применять именно ее. Так что выбор за вами: или использовать старую схему взаимодействия (первая директива), или выбрать более перспективную (вторая директива).

Если у вас вообще нет DNS-сервера (как так получилось?), то чтобы при запуске демон dhcpd не "ругался", добавьте вот такую строку в начало конфигурационного файла:

```
ddns-update-style none;
```

По сути, весь конфигурационный файл DHCP-сервера будет состоять из директивы `ddns-update-style` и блочной директивы `subnet`, описывающей вашу сеть.

Рассмотрим пример объявления сети 192.168.1.0 (листинг 26.1).

Листинг 26.1. Описание сети 192.168.1.0

```
subnet 192.168.1.0 netmask 255.255.255.0 {
# шлюз по умолчанию
        option routers                  192.168.1.1;
# маска сети — этот параметр будет передан всем компьютерам сети
        option subnet-mask              255.255.255.0;
# наш домен
        option domain-name              "firma.ru";
# IP-адрес сервера DNS
        option domain-name-servers       192.168.1.1;

# диапазон IP-адресов: компьютерам нашей сети будут присваиваться
# IP-адреса из этого диапазона
range 192.168.1.10 192.168.1.100;

# время аренды адреса – 7 часов или 25200 секунд
default-lease-time 25200;

# через 28800 секунд (8 часов) IP-адрес будет "отобран",
# если клиент не вернул его через 7 часов
max-lease-time 28800;

}
```

Если в вашей большой сети имеется несколько подсетей, то все подсети (директива `subnet`) должны быть описаны в одной директиве `shared-network`. При этом все общие для подсетей параметры: описание маршрутизаторов, DNS-серверов, доменное имя — выносятся за пределы директив `subnet` (листинг 26.2).

Листинг 26.2. Большая сеть и ее подсети

```
shared-network имя_нашей_сети {
# описываем глобальные для всех подсетей параметры

# домен
    option domain-name                  "firma.ru";
# серверы DNS
    option domain-name-servers          ns1.isp.com, ns2.isp.com;
```

```
# шлюз по умолчанию
    option routers                          192.168.0.1;

# описываем подсети 192.168.1.0 и 192.168.2.0

    subnet 192.168.1.0 netmask 255.255.252.0 {
        range 192.168.1.10 192.168.1.254;
    }
    subnet 192.168.2.0 netmask 255.255.252.0 {
        range 192.168.2.10 192.168.2.254;
    }
}
# конец директивы shared-network
```

26.3. База данных аренды

DHCP-сервер назначает IP-адрес компьютеру не на все время, а только на некоторое, называемое *временем аренды*. По его истечении компьютеру будет назначен другой IP-адрес.

Как мы уже знаем, время аренды регулируется директивами `default-leased-time` и `max-leased-time`, но обычно не нужно изменять значения этих директив, потому что значения по умолчанию вполне приемлемы.

База данных аренды, то есть информация, кому и какой IP-адрес был назначен, находится в файле /var/db/dhcpd.leases. В этом файле содержится следующая информация: уникальный MAC-адрес сетевого адаптера компьютера (аппаратный адрес), назначенный IP-адрес, дата и время окончания аренды и др.

Базу данных аренды нельзя редактировать вручную, ее можно только просматривать.

26.4. Полный листинг конфигурационного файла

Окончательный вариант конфигурационного файла для подсети 192.168.1.0 представлен в листинге 26.3.

Листинг 26.3. Окончательный вариант конфигурационного файла DHCP-сервера

```
# схема взаимодействия с DNS
ddns-update-style ad-hoc;

subnet 192.168.1.0 netmask 255.255.255.0 {

# шлюз по умолчанию
```

```
        option routers                  192.168.1.1;
# маска сети — этот параметр будет передан всем компьютерам сети
        option subnet-mask              255.255.255.0;
# наш домен
        option domain-name              "firma.ru";
# IP-адрес сервера DNS
        option domain-name-servers       192.168.1.1;

# диапазон IP-адресов: компьютерам нашей сети будут присваиваться
# IP-адреса из этого диапазона
range 192.168.1.10 192.168.1.100;
}
```

26.5. Привязка к MAC-адресу

В настоящее время, когда настройкой узла занимается DHCP-сервер, узел сети не будет иметь доступа к сети, если не получит настроек от DHCP-сервера.

Со стороны DHCP-сервера можно блокировать доступ нежелательных компьютеров — точнее, разрешить передачу настроек только тем компьютерам, которым нужно. Делается это путем привязки IP-адресов к MAC-адресам (аппаратным адресам сетевых адаптеров). При такой настройке мы убиваем сразу двух зайцев:

- одному и тому же компьютеру (точнее MAC-адресу) будет всегда назначаться один и тот же IP-адрес, что очень удобно, если система статистики подсчитывает трафик по IP-адресу без аутентификации пользователя;
- компьютеры, MAC-адреса сетевых адаптеров которых вы не прописали в конфигурационном файле DHCP-сервера, не будут иметь доступа к сети, потому что не получат сетевых настроек.

Следует, тем не менее, помнить, что защита средствами DHCP-сервера весьма посредственна и может в этом качестве служить лишь дополнительным барьером. Дело в том, что даже если узел не получит сетевые настройки, их можно указать вручную, а узнать их злоумышленнику не составит особого труда.

Кроме средств DHCP-сервера, нужно также контролировать пространство IP-адресов вашей сети. Например, вы с помощью DHCP выделяете своим клиентам адреса из диапазона 192.168.1.50 — 192.168.1.100, но злоумышленник может указать IP-адрес 192.168.1.101. Если дальше никакие средства (ни прокси-сервер, ни брандмауэр) не осуществляют контроль IP-адресов, толку от контроля MAC-адресов не будет.

К тому же MAC-адрес довольно легко подделать:

- в FreeBSD MAC-адрес для интерфейса можно изменить командами (`XX:XX:XX:XX:XX:XX — MAC-адрес`):

```
# ifconfig <имя_сетевого_интерфейса> down
# ifconfig <имя_сетевого_интерфейса> lladdr XX:XX:XX:XX:XX:XX
# ifconfig <имя_сетевого_интерфейса> up
```

- в Linux вместо второй команды используется следующая:
  ```
  # ifconfig <имя_сетевого_интерфейса> hw ether XX:XX:XX:XX:XX:XX
  ```
- в Windows MAC-адрес можно изменить в свойствах сетевой платы — на вкладке **Дополнительно**, свойство **Сетевой адрес** (рис. 26.1).

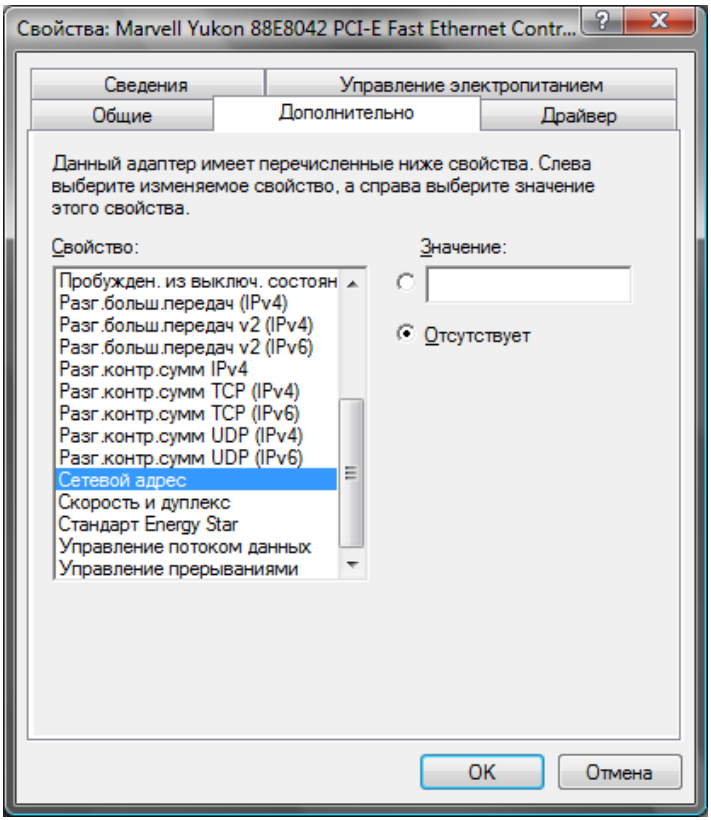


Рис. 26.1. Изменение MAC-адреса сетевого адаптера в Windows

Пояснение

Зачем в книге по BSD приводятся команды изменения MAC-адреса для Linux и Windows? Давайте посмотрим правде в глаза: BSD в основном используется на сервере, но никак не на рабочей станции или домашнем компьютере. Следовательно, потенциальный читатель этой книги является администратором BSD-сервера. А это означает, что ему приходится также настраивать другие компьютеры сети, которые обычно работают как раз под управлением Windows и Linux.

После смены MAC-адреса нужно проверить, установился ли он:

- в FreeBSD — `ifconfig|grep ether`;
- в Linux — `ifconfig -a | grep Hwaddr`;
- в Windows — `ipconfig /all` или командой `getmac` (рис. 26.2).

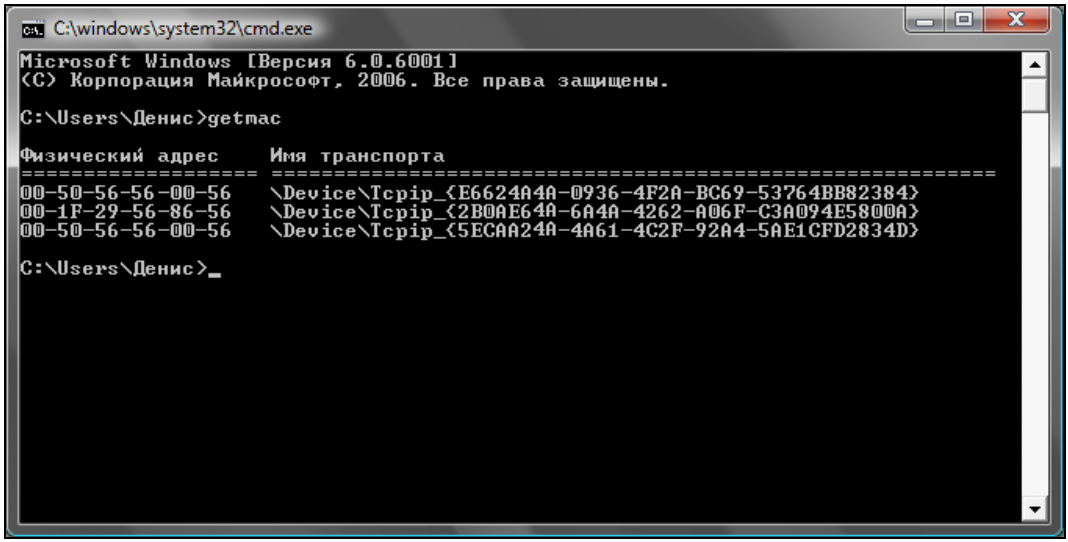


Рис. 26.2. Команда `getmac` в Windows

Спрашивается, откуда злоумышленник узнает, какой MAC-адрес допустим? Ему достаточно подключиться (физически) к вашей сети (что он уже и сделал, раз пытается узнать MAC-адрес, — и вправду, зачем мне MAC-адрес одного из компьютеров Пентагона, если я не собираюсь подключаться к его сети?). Подключившись, он может запустить одну из программ для сбора MAC-адресов, например, TCPNetView (вы ее без проблем найдете в Интернете — программа распространяется бесплатно). Кстати, эта программа полезна и для системного администратора — ведь вам же не хочется вводить команду определения MAC-адреса на каждом компьютере? Вы запускаете программу и получаете MAC-адреса всех подключенных к сети в данный момент компьютеров. Удобно? Я тоже так думаю.

Учитывая все сказанное ранее, можно сделать вывод — защита средствами DHCP-сервера подходит больше для внутреннего контроля, нежели для защиты от взлома сети. Но как дополнительный барьер она вполне уместна.

Если вы таки надумали реализовать привязку IP-адресов к MAC-адресам, добавьте в секцию `subnet` конфигурационного файла каждого компьютера сети следующую конструкцию:

```
host compN {
hardware ethernet xx:xx:xx:xx:xx:xx;
fixed-address IP-адрес;
}
```

Например:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        …
        host comp3 {
```

```
        hardware ethernet 00:40:AA:24:70:2E;
        fixed-address 192.168.1.3;
      }
}
```

26.6. Управление сервером DHCP

Для запуска и останова сервера можно использовать команду `service`:

```
# /usr/local/etc/rc.d/isc-dhcp.sh start
# /usr/local/etc/rc.d/isc-dhcp.sh stop
```

В более старых версиях FreeBSD для управления используется сценарий:

```
# /usr/local/etc/rc.d/isc-dhcpd
```

Чтобы обеспечить автоматический запуск DHCP-сервера, добавьте в файл /etc/rc.conf строки:

```
dhcpd_enable="YES"
dhcpd_ifaces="em0"
```

Первая строка обеспечивает автоматический запуск сервера. Вторая строка содержит список интерфейсов (интерфейсы разделяются пробелами), которые должен "слушать" DHCP-сервер.

26.7. Настройка клиентов

Все клиенты вашей сети (разумеется, кроме серверов сети, которые имеют постоянные IP-адреса) должны быть настроены на автоматическое получение IP-адреса и IP-адресов DNS-серверов. Впрочем, в большинстве случаев рабочие станции и так настроены на автоматическое получение IP-адреса и другой сетевой информации, поэтому настраивать компьютеры вашей локальной сети вам не придется.

Глава 27



DNS-сервер

27.1. Еще раз о том, что такое DNS

Система доменных имен (DNS, Domain Name System) служит для преобразования IP-адресов в доменные имена и обратно. Компьютеру намного проще работать с числами, человеку же легче запомнить символьное имя узла, чем его IP-адрес.

Система DNS имеет древовидную иерархическую структуру (рис. 27.1). Список корневых серверов DNS хранится на каждом DNS-сервере (позже мы узнаем, где именно и как его обновлять).

На рис. 27.1 изображен корень системы DNS, домены первого уровня (ru, com, org) и домен второго уровня (firma). Доменов первого уровня (их еще называют TLD, Top Level Domains) довольно много: com, biz, org, info, gov, net, ws, домены стран (ru, ua, uk, ...) и т. д. Понятно, что доменов второго уровня еще больше, не говоря уже о доменах третьего и последующих уровней.

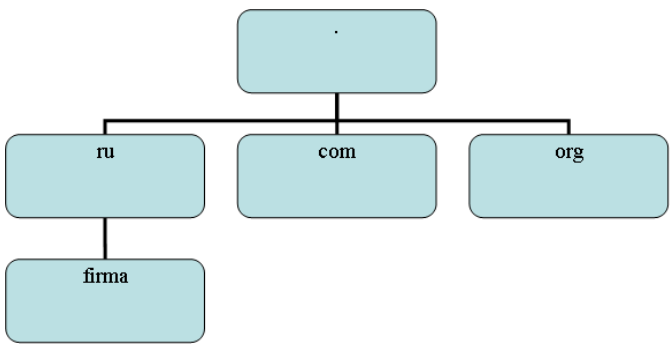


Рис. 27.1. Иерархическая структура DNS

Доменное имя компьютера имеет следующий формат:

```
[имя_компьютера].[домен_N]. ... [домен.TLD]
```

Например,

ftp.sales.firma.ru

При запросе к DNS-серверу доменное имя обрабатывается в доменном порядке. Сначала наш DNS-сервер (это или наш собственный сервер имен, или же сервер

имен нашего провайдера) посылает запрос к DNS-серверу домена **ru** — знает ли он что-нибудь о домене **firma**? Если домен **firma** найден, DNS-сервер домена **ru** сообщает нашему DNS-серверу IP-адрес сервера DNS домена **firma**. Затем наш DNS-сервер обращается к серверу имен домена **firma.ru** с запросом, знает ли он что-либо о домене **sales**. Получив IP-адрес DNS-сервера домена **sales.firma.ru**, мы можем к нему обратиться, чтобы получить IP-адрес компьютера с именем **ftp.sales.firma.ru** (очевидно, это FTP-сервер отдела продаж данной фирмы).

Приведенная схема разрешения доменного имени называется *рекурсивной*, а наш запрос — рекурсивным запросом. Конечно, саму схему я немного упростил, но общий смысл должен быть понятен. Понятно также, что такой запрос занимает довольно много времени и ресурсов, поэтому целесообразно настроить кэширующий сервер DNS, даже если у вас нет собственного домена. Всю "грязную" работу (то есть рекурсивные запросы) будут делать серверы DNS провайдера, а наш сервер будет только кэшировать результаты запросов — так можно повысить скорость разрешения доменных имен и, следовательно, ускорить работу Интернета в целом. Поэтому кэширующий сервер можно установить не только на шлюзе, но и на домашнем компьютере, где он также будет с успехом выполнять свою функцию.

Настройку сервера DNS мы начнем именно с кэширующего сервера DNS *(см. разд. 27.4)*. Во-первых, он настраивается проще, чем полноценный сервер DNS. Во-вторых, в процессе его настройки мы познакомимся с основными конфигурационными файлами, и при настройке полноценного DNS-сервера нам будет легче. В-третьих, не всегда есть необходимость настраивать полноценный DNS-сервер — у вас может быть локальная сеть с выходом в Интернет, но у нее не обязательно должен быть свой собственный домен.

27.2. Запуск DNS-сервера

Прежде чем настраивать DNS-сервер, рассмотрим порядок его запуска. DNS-сервер в BSD установлен по умолчанию, и, чтобы он мог быть запущен, следует добавить в файл /etc/rc.conf строку:

```
named_enable="YES"
```

Из соображений безопасности DNS-сервер по умолчанию запускается в chroot-окружении. Если вы собираетесь запускать его обычно (то есть без chroot-окружения), добавьте в файл rc.conf строку:

```
named_chrootdir=""
```

После этого запустить DNS-сервер можно будет следующей командой:

```
# /etc/rc.d/named start
```

Но сразу запускать установленный по умолчанию DNS-сервер не стоит — перед запуском его надо настроить.

> ***Примечание***
>
> Следует иметь в виду, что наш сервер DNS имеет двойное название: программный продукт в целом называется BIND (Berkley Internet Nameserver Deamon), а основной демон — named. Почему так — никогда не интересовался, да и на конечный результат это никак не влияет.

Все файлы конфигурации сервера BIND находятся в каталоге /etc/namedb, хотя на самом деле это лишь ссылка на каталог /var/named/etc/namedb. Основным файлом конфигурации является файл named.conf, как раз в каталоге /var/named/etc/namedb и находящийся. Сам демон DNS, как уже отмечалось, называется named и находится в каталоге /usr/sbin/. Для управления DNS-сервером, как уже было продемонстрировано ранее, используется сценарий /etc/rc.d/named.

Перед настройкой DNS-сервера сети нужно сказать несколько слов о файле /etc/resolv.conf. В этот файл заносятся директивы `nameserver` — каждая из них содержит только один IP-адрес сервера DNS. Всего может быть указано до четырех DNS-серверов. Для сервера сети, на котором мы разворачиваем DNS-сервер, файл /etc/resolv.conf следует отредактировать так:

```
search ваш_домен
nameserver 127.0.0.1
nameserver <IP-адрес-DNS-вашего-провайдера>
```

Указанные команды говорят о том, что наш сервер в качестве DNS-сервера будет использовать или самого себя (адрес `127.0.0.1`) или DNS-сервер провайдера. А вот на рабочих станциях файл resolv.conf должен выглядеть так:

```
search ваш_домен
nameserver 192.168.1.1
nameserver 192.168.1.2
```

Здесь `192.168.1.1` — адрес первичного, а `192.168.1.2` — адрес вторичного DNS-сервера.

Вот теперь можно приступить к настройке DNS-сервера, осуществляемой, как и практически все настройки в BSD, редактированием файла конфигурации.

27.3. Файл конфигурации named.conf

Файл конфигурации named.conf в FreeBSD просто огромен, даже без комментариев он занимает шесть страниц текста, и приводить его в книге нет смысла — вы и так сможете увидеть этот файл в своей системе. Тем не менее, познакомиться с русскими комментариями к опциям файла named.conf весьма полезно, поэтому я выложил его на своем сайте (в кодировке UTF-8): **http://dkws.org.ua/files/named.conf**.

А сокращенный вариант файла named.conf представлен в листинге 27.1.

Листинг 27.1. Файл named.conf (сокращенный вариант)

```
// Параметры сервера
options {
directory "/etc/namedb";
};
// Наша зона (преобразование имен в IP-адреса)
zone "example.com" {
type master;
```

```
file "master/example.com";
};
// Преобразование IP-адресов в имена
zone "1.168.192.in-addr.arpa" {
type master;
file "master/1.168.192.in-addr.arpa";
};

// Зона подсказок (список корневых серверов находится в
// файле named.root)
zone "." {
type hint;
file "named.root";
};
// Преобразование имени localhost
zone "0.0.127.in-addr.arpa" {
type master;
file "master/localhost.rev";
};
```

В этом конфигурационном файле есть все, что нужно для нормальной работы сервера — блок `options` с параметрами сервера, описание нашей зоны, описание зоны с подсказками (файл named.root).

Примечание

Понятие о зонах (`zone`) и описание опций параметра `type` приводятся в *разд. 27.5*.

Как видно из листинга 27.1, наш базовый файл named.conf стал существенно короче. Вот с ним и будем работать — постепенно наращивать его до нужного нам размера.

27.4. Кэширующий сервер DNS

Что же такое кэширующий сервер DNS? Наверняка все мы знакомы с так называемыми "ускорителями" Интернета — программами, якобы помогающими сделать Интернет намного быстрее. Второе название этих программ — оптимизаторы Интернета. Как правило, это Windows-программы, которые распространяются за определенную плату в Интернете. Иногда их даже можно скачать бесплатно. В первом случае, если программа распространяется за деньги, "ускоритель" Интернета ничего вообще не делает. Он запускается, пользователь устанавливает параметры, но на самом деле никакого ускорения не происходит. Просто кто-то таким не очень честным образом зарабатывает деньги. Во втором случае, когда программа распространяется бесплатно, также не наблюдается никакого ускорения, а наоборот, заметны падение скорости и повышенный расход трафика. Почему? Да по-

тому что "оптимизаторы" Интернета в большинстве случаев являются вирусами-троянами. Пользователи добровольно устанавливают программу, которая потом передаст секретную информацию (например, ключи от электронного кошелька) злоумышленнику. Помните, что бесплатный сыр — только в мышеловке.

Настоящего ускорения Интернета можно добиться установкой кэширующего сервера DNS. Впрочем, не нужно ожидать, что ваш Интернет будет работать на 70, а то и на все 100% быстрее, как это обещают оптимизаторы-вирусы. Некоторое ускорение достигается за счет:

- сокращения времени разрешения доменных имен — поскольку в нашей сети будет свой DNS-сервер, ответы на запросы о разрешении доменных имен станут приходить от локального сервера, а не от загруженного DNS-сервера провайдера;
- определенной экономии трафика — поскольку локальный трафик не будет учитываться, чего не скажешь о трафике между вами (вашей сетью) и провайдером.

Конфигурационный файл named.conf кэширующего сервера DNS представлен в листинге 27.2.

Листинг 27.2. Файл конфигурации /etc/namedb/named.conf

```
// Параметры сервера
options {
directory "/etc/namedb";
};

logging {
category lame-servers { null; };
category cname { null; };
};


forward first;
forwarders {
// IP-адреса DNS-серверов провайдера.
// Адреса разделяются точкой с запятой
}

zone "." {
type hint;
file "named.root";
};

zone "0.0.127.in-addr.arpa" in {
type master;
file "master/localhost.rev";
};
```

В основном конфигурационном файле прописываются корневая и локальная зоны. Локальная зона служит для преобразования имени `localhost` в IP-адрес 127.0.0.1 и наоборот. Корневая зона содержит список корневых серверов DNS.

Файл localhost.rev представлен в листинге 27.3.

Листинг 27.3. Файл localhost.rev

```
$TTL 3600
@ IN SOA ns1.example.com. root. example.com. (
2006101000    ; Серийный номер
3600          ; Обновление
900           ; Повтор
3600000       ; Истечение срока
3600 ) ; Минимальное TTL

IN     NS ns1.example.com.
1      IN PTR localhost.example.com.
```

Отдельного разговора заслуживают *форвард-серверы*. Напомню, что мы сейчас создаем кэширующий сервер, позволяющий ускорить процесс разрешения доменных имен. Но можно ускорить работу самого сервера, указав *форвард-серверы*. В обычном режиме наш сервер сам формирует кэш, но т. к. сеть у нас относительно небольшая, кэш будет формироваться долго, а насколько долго — зависит от количества запросов, поступающих от клиентов сети. Если вы установили кэширующий сервер только для обслуживания своего компьютера, то сначала вообще не почувствуете никакой разницы. Ведь серверу, прежде чем добавить IP-адрес в кэш, нужно его разрешить. И только при втором обращении к доменному имени его IP-адрес будет получен из кэша. А ускорение на начальном этапе может быть обеспечено обращением к кэшу форвард-серверов. Как правило, форвард-серверами выступают серверы провайдера, уже сформировавшие довольно большой кэш, который мы можем использовать.

Все, что нужно для подключения форвард-сервера — это добавить его IP-адрес в блок `forwarders` конфигурационного файла named.conf:

```
forwarders {
         # все запросы будут переадресованы к DNS-серверу
         # провайдера 192.168.99.1
         # если с этим сервером что-то случится, то локальный сервер
         # попробует найти ответ в своем кэше или обратится к другим
         # DNS-серверам, которые указываются в /etc/resolv.conf
              192.168.99.1;
       };
```

Параметр `forwarders` задает заключенный в фигурные скобки список IP-адресов, соответствующих DNS-серверам, которым наш DNS-сервер будет переад-

ресовывать запросы, вместо того, чтобы отвечать на них самому. IP-адреса перечисляются через точку с запятой.

Кроме параметра `forwarders` весьма полезен параметр `forward`, который может принимать следующие значения:

- `only` — наш DNS-сервер никогда не должен предпринимать попыток обработать запрос самостоятельно;
- `first` — наш сервер должен пытаться сам обработать запрос, если указанные далее параметром `forwarders` сервера DNS не были найдены.

Использование параметра `forward` лишено смысла без использования параметра `forwarders`. Параметр `forward` обычно следует указывать до параметра `forwarders`:

```
forward first;
       forwarders {
               192.168.99.1;
               192.168.99.2;
       };
```

А адреса форвард-серверов обычно находятся в файле /etc/resolv.conf.

Вот вроде бы и все, но перед запуском сервера отмечу, что поскольку мы создавали кэширующий сервер, то в его конфигурационном файле отсутствует блок `controls {}`. Пустой или отсутствующий блок `controls{}` необходим для того, чтобы named не обращал внимания на отсутствие ключа `rndc.key`, который нужен для программы удаленного управления сервером — `rndc`. Правда, это не вполне корректно, поскольку для останова сервера придется использовать команду `killall named`, но для нас это не существенно, поскольку мы не будем часто его останавливать.

После редактирования конфигурационных файлов сервера не забудьте его перезапустить.

Теперь осталось в файле /etc/resolv.conf прописать IP-адрес собственного сервера DNS. То же самое нужно сделать на всех остальных компьютерах сети:

```
domain firma.ru
# IP адрес или 127.0.0.1
nameserver 127.0.0.1
# или IP-адрес DNS-сервера — для остальных компьютеров сети
nameserver 10.0.0.1
```

Протестировать настройки можно с помощью программы `nslookup`:

```
# nslookup yandex.ru
```

Server: localhost.firma.ru
Address: 127.0.0.1
Non-authoritative answer:
Name: yandex.ru
Address: 213.180.216.200

Если вы получили подобный ответ, то это означает, что наш сервер работает нормально. Обратите внимание, что ответ пришел не от DNS-сервера провайдера, а от нашего локального сервера.

27.5. Полноценный DNS-сервер

Прежде чем перейти к настройке полноценного сервера DNS, определимся с понятием *зоны*, поскольку полноценный DNS-сервер обслуживает одну или несколько зон. Ошибочно считать зоной обслуживаемый домен — это не так. Давайте разберемся, в чем разница. Домен — это группа компьютеров с одинаковой правой частью доменного имени. Рассмотрим домен **firma.ru**. Компания, которой принадлежит этот домен, довольно большая, поэтому для каждого подразделения ей пришлось организовать свой домен: **sales.firma.ru**, **dev.firma.ru**, **orders.firma.ru** и т. п. Для управления всем доменом **firma.ru** и всеми его поддоменами мы можем использовать или один-единственный DNS-сервер, или же создать независимые серверы для каждого поддомена (или только для некоторых поддоменов). Например, основной сервер будет обслуживать домены **firma.ru** и **sales.firma.ru**, а дополнительный сервер — домены **dev.firma.ru** и **orders.firma.ru**. В таком случае домены **firma.ru** и **sales.firma.ru** образуют одну зону, а домены **dev.firma.ru** и **orders.firma.ru** — другую. Другими словами, *зона* — это часть домена, управляемая определенным DNS-сервером. Зона, которая содержит домены низшего уровня, называется *подчиненной зоной* (subordinate zone).

Вот теперь можно приступить к настройке сервера. Первым делом настроим удаленное управление сервером. Для этого нужно прописать секцию `controls`, которую мы не включали в предыдущих примерах. Выполните команду:

```
# /usr/sbin/rndc-confgen > rndc.conf
```

Откройте файл rndc.conf в любом текстовом редакторе. Выделите и скопируйте две директивы:`key` и `controls`:

```
key "rndc-key" {
      algorithm hmac-md5;
      secret "ключ";
};
controls {
# разрешаем "удаленное" управление только с локального компьютера
      inet 127.0.0.1 port 953
      allow { 127.0.0.1; } keys { "rndc-key"; };
};
```

Скопированный блок текста вставьте в самое начало файла named.conf. Понятно, что из него нужно удалить пустую директиву `controls`, если она в этом файле имеется.

При настройке кэширующего сервера DNS мы в его конфигурационном файле описали две зоны: корневую и локальную. Теперь нам нужно описать еще две зоны: прямого и обратного преобразования, которые и будут обслуживать наш домен. Добавьте в файл конфигурации named.conf строки:

```
// Указываем реальное имя домена, а не example.com
zone "firma.ru"  {
       type master;
```

```
        file "firma.ru";
        notify no;
};

zone "1.0.0.10.in-addr.arpa" {
        type master;
        file "10.0.0.1";
        notify yes;
}
```

Здесь параметр `type` задает тип зоны:

- `master` — мастер-зона, главная зона. Наш сервер является главным сервером для этой зоны и на нем хранится мастер-копия зоны — файл firma.ru, который задается параметром `file` и редактируется администратором вручную;
- `slave` — подчиненная зона, копия зоны, хранимая на вторичном сервере DNS. Через некоторое время вторичный сервер DNS получает информацию о зоне от главного (master) сервера. Файл подчиненной зоны, как правило, не редактируется вручную, а получается автоматически. О создании вторичного сервера DNS мы поговорим отдельно;
- `hint` — зона подсказок, используется для задания имени файла, содержащего список корневых серверов DNS;
- `stub` — напоминает `slave`, но не принимает NS-записи мастер-зоны;
- `forward` — используется для пересылки запросов на другой сервер, указанный в блоке `forwarders`.

Файл firma.ru (он должен находиться в каталоге, заданном директивой `directory`) используется для прямого преобразования, то есть для преобразования доменных имен в IP-адреса. В листинге 27.4 представлен пример этого файла.

Листинг 27.4. Пример файла прямого преобразования

```
@       IN      SOA     server.firma.ru.      hostmaster.firma.ru. (
                20040603        ; серийный номер (можно узнать в
                                ; файлах с примерами)
                3600            ; обновление каждый час
                3600            ; повтор каждый час
                3600000; время хранения информации 1000 часов
                3600            ; TTL записи
)
        IN NS           server.firma.ru.
        IN A            10.0.0.1
        IN MX           100     server.firma.ru.
www     IN CNAME        server.firma.ru.
ftp     IN CNAME        server.firma.ru.
mail    IN CNAME        server.firma.ru.
```

```
c2      IN A          10.0.0.2
c3      IN A          10.0.0.2
localhost.    IN A    127.0.0.1
```

Разберемся, что означают записи этого файла:

- прежде всего обратите внимание, что в конце каждого доменного имени ставится точка — это для того, чтобы сервер не приписывал имя домена (`firma.ru`) к доменному имени. Чтобы не писать каждый раз имя домена, можно просто указывать имя компьютера (`server` вместо `server.firma.ru`), но тогда не нужно ставить точку в конце доменного имени;
- запись `IN SOA` описывает начало полномочий (Start Of Authority, SOA). Первое имя после `SOA` — это имя данного компьютера (на котором запущен DNS-сервер). В нашем случае это `server.firma.ru`. Затем следует e-mail администратора сервера, но поскольку символ `@` зарезервирован, вместо него ставится точка. Остальные элементы записи `SOA` прокомментированы в листинге;
- запись `IN NS` задает имя сервера доменных имен, а запись `IN A` — его IP-адрес. Запись `IN MX` используется для задания почтового сервера. Как мы видим, в роли почтового сервера выступает все тот же наш `server.firma.ru`. 100 — это приоритет почтового сервера. Приоритет указывается, если организовано два или более почтовых сервера. Чем меньше число, тем выше приоритет:

  ```
  IN MX       100   mail1
  IN MX       150   mail2
  ```

- запись `IN CNAME` служит для определения канонических имен, то есть псевдонимов. Как мы видим, к нашему серверу `server.firma.com` можно обратиться по следующим именам: `www.firma.ru`, `ftp.firma.ru`, `mail.firma.ru`;
- далее описаны два компьютера: `c2.firma.ru` (мы не ставили точку после `c2`, поэтому `firma.ru` сервер допишет автоматически) и `c3.firma.ru` с IP-адресами 10.0.0.2 и 10.0.0.3 соответственно;
- последняя запись — это определение имени `localhost`, желательно не забыть о нем.

В листинге 27.5 представлен файл обратного преобразования, использующийся для преобразования IP-адресов в доменные имена.

Листинг 27.5. Пример файла обратного преобразования

```
@       IN      SOA     server.firma.ru.      hostmaster.firma.ru. (
                20040603        ; серийный номер (можно узнать в файлах с примера-
ми)
                3600            ; обновление каждый час
                3600            ; повтор каждый час
                3600000; время хранения информации 1000 часов
                3600            ; TTL записи
)
```

```
@       IN      NS      server.firma.ru
1       IN      PTR     server.firmaru
2       IN      PTR     c2.firma.ru
3       IN      PTR     c3.firma.ru
```

В этом файле, если вы успели заметить, можно полностью не указывать IP-адрес, но требуется полностью указывать доменное имя (точки в конце доменного имени не нужны). Если же вам хочется указать IP-адрес полностью, тогда следует указывать его в обратном порядке, например:

```
2.0.0.10        IN      PTR     c2.firma.ru
```

Вот, практически, и все. Можно в целях защиты сервера добавить в блок `options` конфигурационного файла named.conf директиву `allow-query`:

```
allow-query {
10.0.0.0/24;
localhost;
}
```

Блок `allow-query` разрешает запросы к серверу только узлам подсети `10.0.0.0` и от узла `localhost`. Узлы других подсетей не смогут использовать наш сервер. Когда вы настраиваете DNS-сервер, который будет работать в локальной сети (обслуживать только клиентов нашей локальной сети), то, по большому счету, блок `allow-query` вам не нужен. Но если вы настраиваете DNS-сервер провайдера или же сервер, работающий в сети с реальными IP-адресами, то директива `allow-query` просто необходима, чтобы "чужие" узлы не смогли использовать наш сервер.

После редактирования конфигурационных файлов сервера нужно перезапустить сервис named:

```
# /etc/rc.d/named restart
```

27.6. Вторичный DNS-сервер

В идеале для поддержки домена должно быть выделено два сервера: первичный и вторичный. Вторичный используется для подстраховки — если вдруг с первичным что-то случится (например, банальная перезагрузка администратором).

Вторичный сервер DNS задается аналогично первичному, но несколько иначе описывается зона домена:

```
zone "firma.ru"  {
       type slave;
       file "firma.ru";
       masters { 10.0.0.1; };
};
```

Как видим, устанавливается тип сервера: подчиненный (`slave`), а в блоке `masters` описываются первичные серверы (у нас он один).

В файл конфигурации первичного сервера необходимо добавить директиву `allow-transfer`, в которой следует указать DNS-серверы, которым разрешен трансфер зоны, то есть все вторичные серверы:

```
options {
...
allow-transfer { 10.0.0.2; };
}
```

27.7. Обновление базы данных корневых серверов

Чтобы база данных корневых серверов всегда была актуальной, ее нужно регулярно обновлять. Получить ее можно с адреса **ftp://ftp.internic.net/domain/named.root**, а обновить — с помощью трех следующих команд (да, именно трех, потому что первые две создают резервную копию предыдущего файла named.root):

```
# cd ~
# cp /etc/namedb/named.root named.root.backup
# fetch ftp://ftp.internic.net/domain/named.root
# cp named.root /etc/namedb/named.root
# /etc/rc.d/named restart
```

Листинг файла named.root из соображений экономии места в книге приводить не стану — вы всегда сможете скачать его самую последнюю версию с узла **ftp://ftp.internic.net/**.



Глава 28

Брандмауэр и шлюз

28.1. Что такое брандмауэр?

Брандмауэр (он же firewall, бастион, межсетевой экран) предназначен для защиты внутренней сети (или всего одного компьютера, напрямую подключенного к Интернету) от вторжения извне. С помощью брандмауэра вы можете контролировать доступ пользователей Интернета к узлам вашей внутренней сети. Также можно контролировать доступ локальных пользователей к ресурсам Интернета — например, вы можете запретить им посещать определенные узлы с целью экономии трафика.

Прежде чем перейти к настройке межсетевого экрана, определимся с терминологией и, в частности, с понятием *шлюз*. Шлюзом называется компьютер, предоставляющий компьютерам локальной сети доступ к Интернету. Шлюз выполняет как бы маршрутизацию пакетов. Но не нужно путать шлюз с обычным маршрутизатором. Маршрутизатор осуществляет простую пересылку пакетов, поэтому его можно использовать для соединения сетей одного типа, например, локальной и локальной, глобальной и глобальной. А шлюз служит для соединения сетей разных типов, например, локальной и глобальной, как в нашем случае. Конечно, сейчас можно встретить маршрутизаторы с функцией шлюза, но это уже, скорее, аппаратные шлюзы, чем простые маршрутизаторы. Поэтому часто термины "маршрутизатор" и "шлюз" употребляются как синонимы, хотя это не совсем верно.

Сложность в соединении сетей разных типов заключается в различной их адресации. Как мы знаем, в локальной сети обычно используются локальные адреса, которые не допустимы в Интернете, например: 192.168.*.* (сеть класса C), 10.*.*.* (класс A) и 172.16.*.*–172.31.*.* (класс B). Поэтому шлюз должен выполнить преобразование сетевого адреса (NAT, Network Address Translation). Лучше пояснить эту процедуру на примере. Предположим, у нас есть шлюз и локальная сеть с адресами 192.168.*.*. Реальный IP-адрес (который можно использовать в Интернете) имеется только у шлюза, пусть это 193.254.219.1. У всех остальных компьютеров — локальные адреса, поэтому при всем своем желании они не могут обратиться к интернет-узлам. На компьютере, играющем роль шлюза, установлены два сетевых интерфейса. Один из них, пусть ppp0, служит для подключения к Интернету. Его IP-адрес, как мы условились, 193.254.219.1. С локальной сетью этот компьютер соединен через другой сетевой интерфейс — eth0 (сетевую плату) с IP-адресом

192.168.1.1. Это означает, что все узлы нашей локальной сети передают свои запросы на узел 192.168.1.1. Запросы передаются в виде:

```
Назначение: IP-адрес узла Интернета
Источник: адрес компьютера локальной сети, пусть 192.168.1.10
```

Наш шлюз принимает запрос и перезаписывает его так:

```
Назначение: IP-адрес узла Интернета
Источник: 193.254.219.1
```

То есть шлюз подменяет адрес источника, устанавливая в качестве этого адреса свой реальный IP-адрес (поскольку любой интернет-узел не принял бы запрос с локального адреса). Получив ответ от интернет-узла, шлюз направляет его нашему узлу:

```
Назначение: 192.168.1.10
Источник: IP-адрес узла Интернета
```

При этом нашему локальному узлу "кажется", что он получил ответ непосредственно от узла Интернета, когда на самом деле ответ приходит от шлюза.

Теперь, когда мы разобрались с теорией, самое время перейти к практике.

28.2. Перекомпиляция ядра

Прежде всего нужно убедиться, что ядро ОС собрано с поддержкой брандмауэра. Если это не так (а в большинстве случаев это не так), в файл конфигурации ядра нужно добавить строки:

```
options IPFIREWALL
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_FORWARD
options IPDIVERT
options DUMMYNET
# options IPFIREWALL_NAT
```

Здесь первая строка включает брандмауэр, вторая — разрешает прием всех пакетов, третья — добавляет поддержку перенаправления пакетов (опция `fwd` команды `ipfw`). Третья строка также нужна, если вы надумаете установить прозрачный прокси-сервер. Четвертая строка включает поддержку NAT, а пятая — включает в ядро модуль `DUMMYNET`, необходимый для ограничения величины информационных потоков. Эта опция не является обязательной, и вы вполне можете обойтись без нее.

Последняя строка закомментирована — она добавляет в ядро модуль `ipfw_nat` (о нем будет рассказано далее). Если вы добавили в ядро модуль `ipfw_nat`, удалите из файла конфигурации ядра строку `options IPDIVERT` — эти опции не являются не совместимыми, просто я не вижу смысла включать в ядро лишний код.

Процесс перекомпиляции ядра был рассмотрен в *главе 21*, поэтому сейчас мы на нем останавливаться не будем.

> ***Совет***
>
> Если вы в будущем планируете настраивать VPN *(см. главу 35)*, добавьте в конфигурационный файл ядра следующие опции (чтобы лишний раз потом не перекомпилировать ядро):
>
> ```
> options         IPSEC
> options         IPSEC_ESP
> ```

Можно обойтись и без перекомпиляции ядра. Такой вариант подойдет тем администраторам, которым пересобирать ядро совсем не с руки (или по тем или иным причинам невозможно). Добавьте в файл /boot/loader.conf строки:

```
ipfw_load="YES"
#ipfw_nat_load="YES"
ipdivert_load="YES"
dummynet_load="YES"
```

Этими командами будут загружены модули `ipfw` и `ipdivert`.

Далее, чтобы обойтись без перезагрузки компьютера, введите команды:

```
# kldload ipfw
# kldload ipdivert
# kldload ipfw_nat
# kldload dummynet
```

Для организации NAT в современных версиях FreeBSD можно также использовать уже упоминавшийся ранее модуль `ipfw_nat` (NAT на уровне ядра). Тогда прописывать в файле /boot/loader.conf и загружать командой `kldload` модуль `ipdivert` не нужно, не понадобится и демон natd *(см. разд. 28.4)*. Любители экспериментов могут перейти на `ipfw_nat`[1] (кстати, его поддержка появилась еще в FreeBSD 7.0), но не многие администраторы пока решились сделать это.

> ***Примечание***
>
> Согласен, что демон natd и модуль `ipdivert` немного устарели. Зато данное решение проверено временем и работает в любых версиях FreeBSD, можно сказать — классика. Оставляю решение на ваше усмотрение, а здесь мы будем рассматривать модуль `ipdivert` и демон `natd`.

Итак, модули `ipdivert` или `ipfw_nat` каждый загружает по своему усмотрению. Модуль `dummynet`, как уже отмечалось, можно не загружать — он не является обязательным.

28.3. Конфигурация сети

Для настройки шлюза вам понадобятся как минимум два сетевых интерфейса. Какие именно, зависит от способа подключения к Интернету. Один из таких интерфейсов "смотрит" в локальную сеть, а второй — подключен к Интернету. Это самая тривиальная конфигурация, которая рассматривается в многочисленных руководствах по настройке ipfw (так называется брандмауэр в FreeBSD) и книгах по настройке BSD.

Мы нашу конфигурацию усложним. Пусть у нас будут три сетевых платы:

- em0 — внешний интерфейс, подключенный к провайдеру, IP-адрес 109.95.33.1 (IP-адрес взял из головы, не нужно думать, что он мне принадлежит);

[1] Полное руководство по `ipfw_nat` на русском языке можно найти по адресу: **http://www.lissyara.su/articles/freebsd/tuning/ipfw_nat/**.

- em1 — подключен к локальной сети с реальными адресами, IP-адрес 109.95.33.2 (адрес сети 109.95.33.0/28, сетевая маска 255.255.255.240);
- em2 — подключен к локальной сети с локальным адресом 192.168.1.0, адрес этого интерфейса 192.168.1.1.

Такая конфигурация вполне реальна для небольшой корпоративной сети. Да, она выходит за пределы описания сети небольшого офиса, где имеются два сетевых интерфейса (один — Интернет, второй — локальная сеть). Зато эта конфигурация похожа на "боевые условия" — в локальной сети с реальными IP-адресами могут находиться наши серверы: Web-сервер, почтовый сервер и т. д. А локальная сеть 192.168.1.0 — это обычные пользователи, которых может быть много, поэтому реальных IP-адресов на всех не хватит.

Наши задачи:

- защитить шлюз от атак SYNFlood[1] и ICMP Flood[2];
- закрыть доступ во внутреннюю сеть извне по портам 135, 137 и 139 — эти порты используются службой общих файлов и принтеров Windows;
- разрешить пользователям локальной сети работать с электронной почтой, просматривать Web-страницы и использовать FTP-серверы Интернета;
- запретить доступ к корпоративному серверу (запущены сервисы WWW и FTP, адрес 109.95.33.5), который используется только в пределах нашей локальной сети;
- открыть к шлюзу доступ узлу администратора — он понадобится для удаленной настройки шлюза в будущем.

Казалось бы, все просто. Осталось только реализовать нашу конфигурацию.

Ну, а после настройки шлюза можно приступить к настройке прокси-сервера Squid, который следует "прикрутить" к интерфейсу em2 *(см. главу 29)*.

28.4. Редактирование файла /etc/rc.conf

Добавим в /etc/rc.conf строки, задающие параметры сетевых интерфейсов:

```
# Параметры внешнего интерфейса, их нужно уточнить у провайдера
ifconfig_em0="inet 109.95.33.1 netmask 255.255.255.240"
ifconfig_em0="inet 109.95.33.2 netmask 255.255.255.240"

# Параметры внутреннего интерфейса, должны знать вы
ifconfig_em2="inet 192.168.1.1 netmask 255.255.255.0"
```

[1] SYNFlood (SYN-флуд) — одна из разновидностей сетевых атак типа отказ от обслуживания, которая заключается в отправке большого количества SYN-запросов (запросов на подключение по протоколу TCP) в достаточно короткий срок. — *Из Википедии.*

[2] ICMP Flood (ping-флуд, от англ. ping-flood), дословно: наводнение пакетами ping — тип атаки на сетевое оборудование, ставящий своей целью отказ в обслуживании. — *Из Википедии.*

Поскольку мы сейчас редактируем файл /etc/rc.conf, то, чтобы не отвлекаться на это в дальнейшем, добавим в него следующие строки:

```
# Шлюз провайдера
defaultrouter="109.95.34.1"

# Имя нашего компьютера
hostname="gate.firma.ru"

# YES = наш компьютер является шлюзом
gateway_enable="YES"

# Защищаем шлюз от TCP SYNFlood и от ICMP Flood
tcp_extensions="NO"
tcp_drop_synfin="YES"
icmp_drop_redirect="YES"
icmp_log_redirect="YES"

# Включаем брандмауэр
firewall_enable="YES"

# Прежде чем раскомментировать одну из этих опций,
# прочитайте данную главу
#firewall_type="OPEN"
#firewall_type="/etc/firewall.my"
# Сценарий с правилами брандмауэра
# firewall_script="/etc/rc.firewall"

# После того как все настроите, установите YES — это
# подавит вывод "лишних" сообщений на экран
firewall_quiet="NO"

# Включаем NAT и указываем внешний интерфейс
natd_enable="YES"
natd_interface="em0"
natd_flags=""
```

Остальные параметры установите по своему усмотрению.

Опять о ipfw_nat

Как уже отмечалось, модуль `ipfw_nat` позволяет обойтись без демона natd. Если вы выбрали `ipfw_nat`, то вместо строк загрузки `natd`, нужно добавить две следующие строки:

```
firewall_nat_enable="YES"
firewall_nat_interface="em0"
```

Параметр `firewall_type` задает тип брандмауэра:

- `OPEN` — брандмауэр будет пропускать все пакеты. В этом случае скрипт брандмауэра указывать не нужно;
- `CLIENT` — брандмауэр будет использоваться только для защиты этого узла, применяется только на рабочих станциях;
- `SIMPLE` — позволяет построить простой шлюз, который защищает внутреннюю сеть от проникновения извне;
- `CLOSED` — разрешает прохождение трафика только через интерфейс lo0, остальной трафик (через другие интерфейсы) запрещен;
- `UNKNOWN` — запрещает загрузку правил из конфигурационного скрипта по умолчанию. Брандмауэр никак не настраивается. Что он будет делать — пропускать трафик или нет, зависит от настроек ядра. Именно поэтому мы добавили вторую опцию (помните: `options IPFIREWALL_DEFAULT_TO_ACCEPT`) в файл конфигурации ядра — чтобы брандмауэр по умолчанию пропускал трафик. Используется по умолчанию;
- `имя_файла` — загружает правила из указанного файла. Каждое правило — это инструкция брандмауэру, что он должен делать. Например, какие пакеты он должен пропускать, а какие — блокировать.

Вы можете выбрать один из этих профилей и вообще не указывать файл правил. По умолчанию конфигурация (набор правил) брандмауэра берется из файла /etc/rc.firewall. Правила для всех профилей указываются в файле rc.firewall. Вы можете установить параметр `firewall_type` так:

```
firewall_type="MY"
```

После этого нужно в файле rc.firewall создать раздел `MY`, в котором прописать ваши правила *(см. разд. 28.5)*. А можно прописать все необходимые правила в отдельном файле *(см. разд. 28.6)* и указать его имя. Рассмотрим далее оба варианта.

28.5. Редактирование файла /etc/rc.firewall

Создать отдельный файл с правилами просто, а вот отредактировать файл rc.firewall — не очень. Нужно знать особенности написания sh-сценариев. В некоторых руководствах авторы даже советуют не редактировать этот файл.

Будем считать, что в файл /etc/rc.conf вы поместили рекомендованную ранее строку:

```
firewall_type="MY"
```

Эта строка означает, что наш набор правил будет называться `MY`. Откройте файл rc.firewall:

```
# ee /etc/rc.firewall
```

Перейдите в самый конец файла и найдите строку:

```
*)
```

Перед этой строкой нужно вставить следующую строку:

```
[Mm][Yy])
```

Это и есть название нашего профиля. Квадратные скобки используются, чтобы задать имя профиля в файле rc.conf любым удобным образом: MY, my, My, mY.

После добавленной нами строки добавьте строки, описывающие правила брандмауэра:

```
# Некоторые переменные
net="109.95.33.0/28"
mask="255.255.255.255.240"

# Разрешаем трафик интерфейсу lo0 — стандартное первое правило
# любого шлюза
${fwcmd} add pass all from any to any via lo0

# Запрещаем фрагментированные пакеты
${fwcmd} add deny icmp from any to any frag

# Разрешаем ICMP-пакеты
${fwcmd} add pass ICMP from any to any

# Разрешаем трафик в пределах локальной сети
${fwcmd} add pass all from any to any via em1

# Разрешаем протокол SMTP (отправка почты, порт 25)
${fwcmd} add pass tcp from any to any 25 out
${fwcmd} add pass tcp from any 25 to any out

# Разрешаем протокол HTTPS
${fwcmd} add pass tcp from any to any 443 out
${fwcmd} add pass tcp from any 443 to any out

# Запрещаем доступ извне к корпоративному серверу компании
${fwcmd} add deny tcp from any to 109.95.33.5 80 in via em0

# Разрешаем протокол HTTP
${fwcmd} add pass tcp from any to any 80 out via em1
${fwcmd} add pass tcp from any 80 to any out via em1

# В локальной сети (с реальными адресами) разрешаем все протоколы
${fwcmd} add allow all from any to any via em1

# Следующие правила разрешают работу DNS
${fwcmd} add pass udp from any to any 53
${fwcmd} add pass udp from any 53 to any

# Разрешаем работу с новостными серверами (порт 119)
${fwcmd} add pass tcp from any to any 119 out via em1
```

```
${fwcmd} add pass tcp from any 119 to any out via em1

# Разрешаем протокол FTP (порты 20 и 21)
# Порт 20 используется для передачи данных
${fwcmd} add pass tcp form any 21 to any
${fwcmd} add pass tcp from any to any 21
${fwcmd} add pass tcp from any 20 to any
${fwcmd} add pass tcp from any to any 20

# Разрешаем протокол POP3
${fwcmd} add pass tcp from any to any 110
${fwcmd} add pass tcp from any 110 to any

# Разрешаем администратору доступ по SSH
# IP-адрес машины администратора 109.95.33.20
${fwcmd} add pass tcp from 109.95.33.2 22 to {isp}
${fwcmd} add pass tcp from {isp} to 109.95.33.2 22

# Ограничиваем трафик с em2 в локальную сеть с реальными
# адресами — em1
${fwcmd} add deny all from 192.168.1.0/24 to ${net}:${mask} via em1
${fwcmd} add deny all from ${net}:${mask} to 192.168.1.0/24 via em1
# Разрешаем порты 25, 110, 53, 80 для локальной сети, подключенной
# к em2. Аналогично, вы можете разрешить другие порты
${fwcmd} add pass tcp from 192.168.1.0/24 to any 25 via em2
${fwcmd} add pass tcp from any 25 to 192.168.1.0/24 via em2
${fwcmd} add pass tcp from 192.168.1.0/24 to any 110 via em2
${fwcmd} add pass tcp from any 110 to 192.168.1.0/24 via em2
${fwcmd} add pass udp from 192.168.1.0/24 to any 53 via em2
${fwcmd} add pass udp from any 53 to 192.168.1.0/24 via em2
${fwcmd} add pass tcp from 192.168.1.0/24 to any 80 via em2
${fwcmd} add pass tcp from any 80 to 192.168.1.0/24 via em2
```

Сохраняем файл и по примеру Microsoft перезагружаем машину.

28.6. Создание отдельного файла правил

Если вы хотите создать отдельный файл с правилами, укажите его в параметре `firewall_type`. В начало файла поместите строки:

```
#!/bin/sh
/sbin/ipfw -f flush
```

Затем добавьте в этот файл все правила — все, что шло после строки [Mm][Yy] в *разд. 28.5*. При этом фрагмент `${fwcmd}` замените командой `/sbin/ipfw` (или можете создать переменную `fwcmd` и оставить все как есть — на ваше усмотрение).



Глава 29

Прокси-сервер

29.1. Зачем нужен прокси-сервер в локальной сети?

С помощью прокси-сервера можно очень эффективно управлять ресурсами своей сети, например, кэшировать трафик (HTTP), "обрезать" баннеры, указывать, какие файлы можно скачивать пользователям, а какие — нет, можно также указать максимальный объем передаваемого объекта и даже ограничить пропускную способность пользователей определенного класса.

Но основная функция прокси-сервера — это кэширование трафика. С его помощью можно сократить кэш браузеров клиентов практически до нуля — он уже не будет нужен, поскольку кэширование станет выполнять прокси-сервер. Дело в том, что когда прокси-сервер выполняет кэширование всех клиентов сети, уже запрошенные ранее кем-то страницы оказываются доступными другим пользователям. Это означает, что если один из клиентов сети зашел на сайт **firma.ru**, то у всех остальных пользователей сети этот сайт будет открываться практически мгновенно, потому что его уже кэшировали.

Даже если у вас всего один компьютер, все равно имеет смысл использовать прокси-сервер, хотя бы для того, чтобы "обрезать" баннеры — так можно сэкономить на трафике, да и страницы начнут открываться быстрее, если многочисленные баннеры не будут грузиться.

Прокси-сервер Squid, рассматриваемый нами здесь, не сложен в настройке, во всяком случае он не сложнее Samba и подобных сетевых сервисов. Для его установки нужно установить порт squid:

```
# cd /usr/ports/www/squid
# make install clean
```

Любители экономить время могут установить Squid из пакетов:

```
# pkg_add -r squid
```

После установки пакета у вас в системе появится новый сервис — squid. Основной конфигурационный файл его — /usr/local/squid/etc/squid.conf. После установки для автоматического запуска добавьте в файл /etc/rc.conf строку:

```
squid_enable="YES"
```

29.2. Базовая настройка Squid

Базовая настройка Squid осуществляется путем редактирования его основного конфигурационного файла /usr/local/squid/etc/squid.conf (листинг 29.1).

Листинг 29.1. Файл /usr/local/squid/etc/squid.conf

```
# Порт для прослушивания запросов клиентов
# задается в формате http_port <порт> или http_port <узел>:<порт>
# Последний случай подходит, если Squid запущен на машине с несколькими
# сетевыми интерфейсами
http_port 192.168.1.1:3128

# адрес прокси провайдера, нужно согласовать с провайдером
# cach_peer proxy.your_isp.com

# Объем оперативки в байтах, который будет использоваться прокси-сервером
# (85 Мбайт). Не устанавливайте более трети физического объема оперативки,
# если данная машина должна использоваться еще для чего-либо.
# Объем можно задать в мегабайтах, но тогда между числом и MB обязательно
# должен быть пробел: cache_mem 128 MB
cache_mem 128 MB

# Где будет размещен кэш.
# Первое число — это размер кэша в Мбайт, не устанавливайте кэш на весь
# раздел. Если нужно, чтобы он занимал весь раздел, отнимите от размера
# раздела 20% и укажите это значение. Например, если раздел 1024 Мбайт,
# то для кэша — только 820 Мбайт; второе — количество каталогов
# первого уровня;
# третье — к-во каталогов второго уровня
# Убедитесь, что на разделе есть достаточно свободного места!
cache_dir ufs /usr/local/squid/cache 1024 16 256
# К этому каталогу нужно установить права доступа:
# chown -R squid /usr/local/squid/cache

# Когда кэш в памяти будет заполнен на 95%, начнется процесс усиленного
# свопа — старые объекты из кэша в памяти будут перемещены на кэш на
# жестком диске, чтобы освободить место для новых объектов
cache_swap_high 95
# Когда кэш будет заполнен на 85%, процесс усиленного свопа прекратится
cache_swap_low 85

# Максимальный размер кэшируемого объекта,
```

```
# если размер объекта превышает указанный здесь, то объект не будет
# сохранен на диске
# maximum_object_size 10240 KB

# Минимальный размер кэшируемого объекта,
# если размер объекта меньше указанного здесь, то объект не будет
# сохранен на диске
# minimum_object_size 0 KB


# Хосты, с которых разрешен доступ к прокси
acl allowed_hosts src 192.168.1.0/255.255.255.0
acl localhost src 127.0.0.1/255.255.255.255

# Разрешенные порты:
acl allow_ports port 80                    # http
acl allow_ports port 21                    # ftp
# SSL-порты
acl SSL_ports port 443 563

# Запрещаем все порты, кроме указанных в allow_ports
http_access deny !allow_ports

# Запрещаем метод CONNECT для всех портов, кроме указанных в
# acl SSL_ports:
http_access deny CONNECT !SSL_ports

# Запретим доступ всем, кроме тех, кому можно
http_access allow localhost
http_access allow allowed_hosts
http_access allow SSL_ports
http_access deny all

# Пропишем пользователей, которым разрешено пользоваться squid
# (ppt, admin):
ident_lookup on
acl allowed_users den admin
http_access allow allowed_users
http_access deny all
```

Базовый конфигурационный файл с успехом выполняет только функцию кэширования, в *разд. 29.3* мы поговорим о более тонкой настройке Squid.

29.3. Практические примеры

29.3.1. Управление доступом

Управление доступом осуществляется с помощью ACL (Access Control List) — списков управления доступом.

Разберемся, как работать с ACL. Создадим список `AllowedPorts`:

```
acl AllowedPorts port 80 8080 3128
```

Итак, имя списка — `AllowedPorts`, тип списка — `port`. Далее мы можем использовать этот список в качестве параметра команды `http_access` для разрешения/запрещения указанных портов:

```
http_access allow AllowedPorts      # разрешение портов
http_access deny AllowedPorts       # запрещение портов
```

Кроме типа `port` часто используются следующие типы списков:

- `proto` — протокол (HTTP или FTP);
- `method` — метод передачи данных (GET или POST);
- `src` — IP-адреса (или диапазоны адресов) клиентов;
- `dst` — IP-адреса/URL сайтов, к которым обращаются клиенты.

Вы также можете создать список узлов, которым разрешен доступ к прокси (файл /etc/squid/allowed-hosts.txt):

```
acl allowed_hosts src "/usr/local/etc/squid/allowed-hosts.txt"
```

Сам файл /etc/squid/allowed-hosts.txt может выглядеть так:

```
# den
192.168.1.2/255.255.255.255
# admin
192.168.1.3/255.255.255.255
```

Отдельный файл использовать удобнее, поскольку при этом не засоряется основной конфигурационный файл.

> ***Внимание!***
>
> Права доступа к файлу allowed-hosts.txt должны быть такими же, как и к файлу squid.conf.

29.3.2. Создание черного списка URL

Теперь попробуем создать черный список URL:

```
acl blacklist url_regex adult
http_access deny blaklist
http_access allow all
```

Данный черный список не пропускает URL, содержащие слово `adult`. По аналогии можно было бы создать отдельный файл и записать в него все "плохие" URL, однако это довольно накладно — проще использовать регулярные выражения.

29.3.3. Отказ от баннеров

С помощью ACL можно отказаться и от баннеров — принцип тот же. Для этого добавьте в файл конфигурации следующие ACL:

```
acl banners urlpath_regex "/usr/local/etc/squid/banners.txt"
http_access deny banners
```

В файл banners.txt нужно внести URL баннерных сетей, например,

```
^http://www.clickhere.ru
^http://banner.kiev.ua
...
```

Создание этого файла пусть будет вашим домашним заданием — все равно все баннерные сети в книге не приведешь.

29.4. Управление прокси-сервером

Для запуска, перезапуска и остановки прокси-сервера следует использовать команды:

```
# /usr/local/etc/rc.d/squid start
# /usr/local/etc/rc.d/squid restart
# /usr/local/etc/rc.d/squid stop
```

29.5. Настройка клиентов

Все браузеры на компьютерах вашей сети нужно настроить на использование порта 3128 (именно этот порт мы установили в конфигурационном файле). На рис. 29.1 изображена настройка браузера Opera.

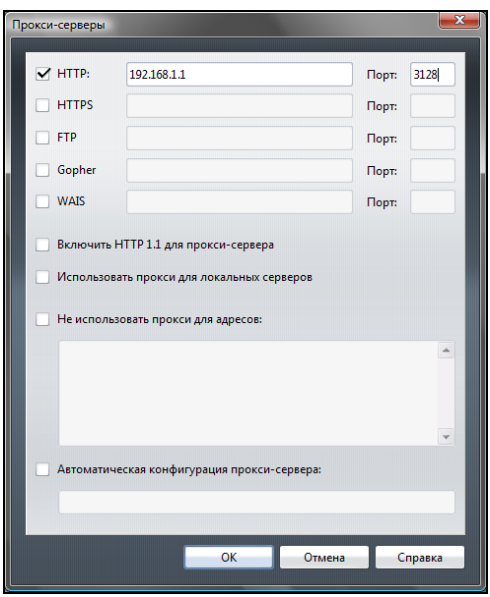


Рис. 29.1. Настройка клиента

Примечание

Настройка прозрачного прокси-сервера описана в *разд. 29.8*.

29.6. Отказ от баннеров с помощью редиректора Rejik

Ранее было показано, как блокировать баннеры средствами Squid. Недостаток данного решения заключается в том, что список баннеров придется постоянно обновлять, иначе толку от нашей блокировки не будет. Чтобы освободить себя от постоянного поиска новых баннерных сетей (и так есть чем заниматься), мы добавим к Squid редиректор Rejik, позволяющий блокировать баннеры по постоянно обновляющимся бан-листам. Другими словами, вам нужно будет один раз установить Rejik, а дальше он все сделает за вас сам.

Настраивается Rejik элементарно просто — при желании можно успеть за 15 минут (если не считать времени сборки самого Rejik). Устанавливаем Rejik:

```
# cd /usr/ports/www/rejik
# make install
```

При компиляции выбираем все доступные опции, цель `clean` не вводим — файлы пока очищать не нужно, позже они нам пригодятся.

Копируем бан-листы в конфигурационный каталог Rejik:

```
# cp /usr/ports/www/rejik/work/banlist /usr/local/rejik
```

Совет

Обновленную версию бан-листов можно скачать с сайта **www.rejik.ru**. На этом же сайте можно получить дополнительную информацию о настройке Rejik.

Теперь нужно скопировать файлы из каталога squid-like-www-en (в нем находятся картинка, которой мы будем заменять баннеры, а также HTML-странички, которые будут выводиться при попытке открытия запрещенных ресурсов):

```
cp -R /usr/ports/www/rejik/work/squid-like-www-en /usr/local/www/rej
```

Внимание!

Web-сервер Apache (или любой другой — использование Apache в данном случае не принципиально) должен быть настроен и запущен *(см. главу 34)*.

Отредактируем файл конфигурации squid.conf, добавив в него две строки:

```
url_rewrite_program /usr/local/rejik/redirector /usr/local/rejik/redirector.conf
url_rewrite_children       15
```

Сохраним файл squid.conf и откроем файл конфигурации redirector.conf. Вполне рабочий вариант этого файла приведен в листинге 29.2.

Листинг 29.2. Файл redirector.conf

```
# Протокол ошибок редиректора
error_log /usr/local/rejik/redirector.err
# Протокол замен: прочитав этот протокол, вы поймете, кто и куда
# хотел "пойти"
change_log /usr/local/rejik/redirector.log

make-cache /usr/local/rejik/make-cache

# Наши IP-адреса
# IP-адрес Web-сервера: 192.168.1.1
work_ip 127.0.0.1/8
work_ip 192.168.1.0/24

<BANNER>
# Баннеры будут заменены картинкой 1x1.gif
ban_dir /usr/local/rejik/banlists/banners
url http://192.168.1.1/rej/1x1.gif
# Протоколы замен (change_log) занимают много места, а особого
# толку от них нет — сначала посмотрите, а как наиграетесь,
# отключите протоколирование:
# log off

<PORNO>
# Страницы с порно будут заменены страницей porno.html
ban_dir  /usr/local/rejik/banlists/porno
url http://192.168.1.1/rej/porno.html
# log off

<AUDIO-VIDEO>
ban_dir /usr/local/rejik/banlists/mp3
url http://192.168.1.1/rej/audio-video.html
# log off

<JS>
ban_dir /usr/local/rejik/banlists/js
url http://192.168.1.1/rej/js.js
# log off
```

Теперь осталось установить необходимые права доступа:

```
# /usr/local/rejik/tools/set-permissions
```

На этом все. Перезапустите Squid и наслаждайтесь работой Rejik.

29.7. Анализатор протоколов Squid

SARG (Squid Analysis Report Generator) — анализатор протоколов Squid, как следует из его названия, анализирует протоколы и генерирует отчеты по ним. В результате мы узнаем, кто и какие ресурсы посещал, сколько мегабайтов скачал. SARG умеет строить довольно красивые графики — по времени суток, дням месяца и т. д. Программа очень маленькая, но весьма полезная. Для подобных целей существуют целые системы статистики, которые работают постоянно и собирают в базу данных информацию о проходящем через шлюз трафике. Но такие системы "пожирают" системные ресурсы, а SARG можно запустить только тогда, когда вам нужно. Например, когда руководство захочет получить от вас отчет об использовании Интернета за месяц. Вы запускаете SARG, он анализирует протоколы Squid (для актуальности статистики нужно, чтобы все пользователи "ходили" в Интернет через Squid) и через некоторое время выдает отчет.

Статистика генерируется в виде HTML-страниц, что облегчает ее чтение. Желательно, чтобы на сервере, где запускается SARG, был установлен Web-сервер — так будет удобнее просматривать статистику. Впрочем, можно скопировать отчеты себе на компьютер, однако боюсь, что вы можете не успеть к планерке — ведь придется скопировать несколько тысяч мелких файлов, которые копируются намного медленнее, чем один большой.

Чтобы решить, нужен вам SARG или нет, можете просмотреть образцы отчетов на сайте разработчиков SARG: **http://sarg.sourceforge.net/squid-reports/index.html**.

Установим SARG:

```
# /usr/ports/www/sarg
# make install
```

После чего отредактируем файл его конфигурации /usr/local/etc/sarg/sarg.conf. Все опции файла тщательно прокомментированы, и вы можете изменить их по своему усмотрению. Самый главный параметр — это `output_dir`, задающий каталог, в который нужно поместить отчет:

```
output_dir /usr/local/www/data/statistic/http_stat
```

Также можно задать язык и кодировку отчета:

```
language Russian_koi8
charset Koi8-r
```

Для запуска SARG следует выполнить команду:

```
# /usr/local/bin/sarg -l squid.log
```

Перед выполнением этой команды нужно перейти в каталог, где находится файл squid.conf, или указать полный путь к этому файлу в команде.

Процесс генерации отчета займет немало времени (особенно на медленных машинах) из-за необходимости записать множество небольших файлов. На современной машине частотой 2 ГГц на обработку журнала размером в 300 Мбайт уходит примерно 5 минут.

29.8. Прозрачные прокси-серверы

Рассмотрим теперь организацию прозрачного прокси-сервера. Прозрачный прокси-сервер не требует настройки клиентов — они даже подозревать не будут, что в сети установлен прокси-сервер.

29.8.1. Установка прокси-сервера OOPS

Компактный прозрачный прокси-сервер OOPS устанавливается из портов:

```
# cd /usr/ports/www/oops
# make install clean
```

При установке следует выбрать опции `PCRE`, `GIGABASE`. Затем нужно перекомпилировать ядро *(см. главу 21)* со следующими опциями:

```
options         IPFIREWALL
options         IPFIREWALL_FORWARD
options         IPFIREWALL_VERBOSE
options         IPFIREWALL_VERBOSE_LIMIT=100
options         IPFIREWALL_DEFAULT_TO_ACCEPT
options         IPDIVERT
```

В конфигурационный файл /etc/rc.d/rc.firewall.local брандмауэра ipfw добавляем правило:

```
add fwd 127.0.0.1,3128 tcp from 192.168.1.0/24 to any dst-port 80 via em0
```

Данное правило и обеспечивает прозрачность прокси. При желании с помощью этого правила можно сделать прозрачным и обычный Squid *(см. разд. 29.8.2)*. Для этого нужно изменить адрес сети (192.168.1.0) и имя интерфейса (em0).

Теперь отредактируем конфигурационный файл /usr/local/etc/oops/oops.cfg. Весь этот файл приводить не буду — он довольно большой. Для выполнения нашей задачи нужно в группу `mynet` добавить опцию `redir_mods transparent`:

```
group mynet {
              ...
redir_mods transparent;
             ....
}
```

Далее следует определить модуль `transparent` и указать порт прозрачного прокси:

```
module transparent {
       myport 3128
       }
```

Теперь надо определить, где должен находиться кэш, и указать его размер (1024 Мбайт):

```
storage {
        path /usr/local/oops/storages/oops_storage;
        size 1024m;
}
```

Автоматический запуск OOPS обеспечивается путем добавления следующей строки в файл /etc/rc.conf:

```
oops_enable="YES"
```

Создадим хранилище кэша:

```
# oops -z -c /usr/local/etc/oops/oops.cfg
```

И запустим прокси:

```
# /usr/local/etc/rc.d/oops start
```

Вот и все. Можете наслаждаться компактным прокси-сервером.

29.8.2. Прозрачный Squid

Чтобы сделать прозрачным Squid, необходимо добавить в файл squid.conf одну из двух строк:

```
http_port 3129 transparent
http_port 3129 intercept
```

Первая строка добавляется, если у вас старый Squid (до версии 3.1), а вторая — если вы используете самую последнюю версию Squid (3.1). При сборке Squid нужно включить опции `SQUID_IPFW`, `SQUID_PF` и `SQUID_IPFILTER` (набор опций зависит от используемого вами брандмауэра). Если вы их не включили, Squid придется пересобрать (рис. 29.2).

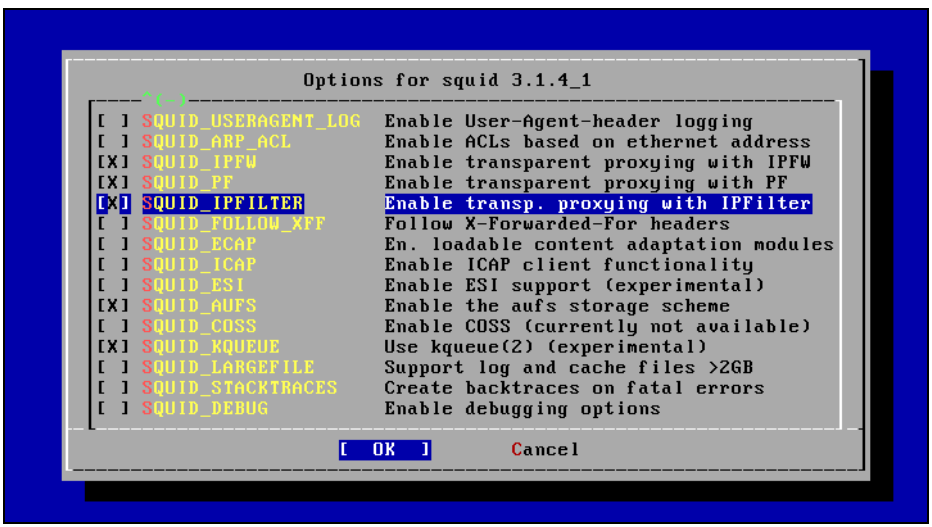


Рис. 29.2. Сборка squid

После настройки Squid нужно добавить в конфигурационный файл /etc/rc.d/rc.firewall.local брандмауэра ipfw уже упомянутое ранее *(см. разд. 29.8.1)* правило, обеспечивающее прозрачность прокси:

```
add fwd 127.0.0.1,3128 tcp from 192.168.1.0/24 to any dst-port 80 via em0
```

29.8.3. Проблемы с прозрачным Squid

Настройка прозрачного прокси-сервера требует правильной настройки ядра, брандмауэра, NAT. Все это уже рассматривалось в книге, но иногда, чтобы получить результат, приходится собрать воедино и заново рассмотреть уже изложенный материал. Так что данный раздел я добавил в книгу практически за один день до отправки рукописи в издательство.

Итак, у вас не получается настроить Squid. Вроде бы и брандмауэр настроили, и NAT работает, и пользователи могут "в Интернет ходить", да и сам Squid работает, но не хочет становиться прозрачным.

Попробую сейчас ответить на извечный вопрос "Что делать?". Прежде всего надо проверить все конфигурационные файлы, от конфигурационных файлов ядра до конфигурационных файлов системы.

Начнем с ядра. В *главе 28* было сказано, что для организации брандмауэра можно обойтись и без перекомпиляции ядра — нужно только вызвать нужные модули. И чтобы не запутать вас еще больше, приведу фрагмент конфигурационного файла ядра, относящийся к брандмауэру (листинг 29.3). При желании можете добавить потом необходимые модули, но я все же советую пересобрать ядро. Да, потеря времени обеспечена, но потом это вам окупится.

Листинг 29.3. Фрагмент файла конфигурации ядра

```
# Enable IP Firewall
options   IPFIREWALL
options   IPFIREWALL_VERBOSE
options   IPFIREWALL_VERBOSE_LIMIT=100
options   IPFIREWALL_DEFAULT_TO_ACCEPT
options   IPFIREWALL_FORWARD
options   IPFIREWALL_NAT
options   IPDIVERT
options   TCPDEBUG
options   LIBALIAS
options   DUMMYNET
```

Поскольку мы не знаем, на основе модуля `ipfw_nat` или демона natd организована ваша конфигурация, включаем обе опции NAT: `IPFIREWALL_NAT` и `IPDIVERT`. После перекомпиляции ядра нужно перезагрузить компьютер и добавить следующие строки в файл /etc/rc.conf:

```
firewall_enable="YES"
firewall_script="/etc/rc.myfw"
```

Строку `firewall_nat_enable="YES"` добавлять в файл /etc/rc.conf не следует, поскольку в ее присутствии почему-то и могут возникнуть проблемы с прозрачным Squid (во всяком случае, возникли у меня).

Далее нужно отредактировать сценарий /etc/rc.myfw (листинг 29.4). В этом файле будут содержаться и правила брандмауэра, и настройки для natd, поэтому-то отдельно вызывать natd из файла rc.conf нет необходимости. Приведенный сценарий — рабочий, вам нужно изменить только имена интерфейсов и их адреса. После тестирования желательно добавить правила брандмауэра, поскольку сейчас для упрощения примера они сведены к минимуму, — все, что не касается прозрачного прокси, в этот сценарий не включено.

Листинг 29.4. Файл /etc/rc.myfw

```
#!/bin/sh —
# Пути к исполнимым файлам ipfw и natd
fwcmd="/sbin/ipfw -q"
natcmd="/sbin/natd"

# Внутренний и внешний интерфейсы
intif="em1"                  # внутренний интерфейс
intip="192.168.1.0/24"       # внутренний IP-адрес

extif="em0"                  # внешний интерфейс
extip="94.94.94.94"          # внешний IP-адрес

${natcmd} -n ${extif} -p 8668

${fwcmd} -q flush
${fwcmd} add check-state

${fwcmd} add pass all from me to me

${fwcmd} add fwd 127.0.0.1,3128 tcp from ${intip} to any http,https via ${in-
tif}
${fwcmd} add divert 8668 all from ${intip} to any via ${extif}
${fwcmd} add divert 8668 all from any to me via ${extif}
```

Теперь перепроверим конфигурационный файл Squid (листинг 29.5). Поскольку все опции конфигурационного файла squid.conf уже были описаны в этой главе ранее, файл конфигурации приводится без комментариев. В нем обязательно должна быть первая строка — она и обеспечивает прозрачность. То есть, если вы не укажете опцию `transparent`, прокси не будет прозрачным.

Представленный в листинге 29.5 конфигурационный файл squid.conf не полон — это только его начало. После приведенных строк идет описание ACL (порты, пользователи) — вряд ли вам будет интересна конфигурация моей сети (тем более, что как работать с ACL, было показано в *разд. 29.1*).

Листинг 29.5. Конфигурационный файл squid.conf

```
http_port 3128 transparent
maximum_object_size 16384 KB
cache_mem  128 MB
cache_dir ufs /usr/local/squid/cache 40000 16 256
coredump_dir /usr/local/squid/cache
access_log /usr/local/squid/logs/access.log squid
cache_access_log /usr/local/squid/logs/access.log
cache_log /usr/local/squid/logs/cache.log
cache_store_log /usr/local/squid/logs/store.log
pid_filename /usr/local/squid/logs/squid.pid
logfile_rotate 10
...
```



Глава 30

FTP-сервер

30.1. Зачем нужен FTP?

Сервер FTP (File Transfer Protocol) используется для обмена файлами между пользователями Интернета. Принцип работы FTP следующий: на FTP-сервере размещается какой-нибудь файл. Пользователи Интернета с помощью FTP-клиента (в любой операционной системе имеется стандартный FTP-клиент — программа ftp) подключаются к FTP-серверу и скачивают этот файл.

Права FTP-пользователя определяются администратором FTP-сервера. Одни пользователи могут загружать на сервер файлы в свои личные каталоги, другие имеют полный доступ к FTP-серверу (могут загружать файлы в любые каталоги — как правило, это администраторы FTP-сервера), третьи могут только скачивать публично доступные файлы. Третья группа пользователей — самая большая. Это так называемые *анонимные пользователи*. Чтобы не создавать учетную запись для каждого анонимного пользователя, все они работают под так называемой *анонимной учетной записью*, когда вместо имени пользователя указывается имя `anonymous`, а вместо пароля — адрес электронной почты пользователя.

В локальной сети для обмена файлами можно использовать сервер Samba, имитирующий работу рабочей станции под управлением Windows, в Интернете же для обмена файлами следует использовать только FTP-сервер. С другой стороны, ничего не мешает вам организовать FTP-сервер для обмена файлами внутри локальной сети — это дело вкуса и предпочтений администратора.

Все необходимое для организации FTP-сервера программное обеспечение входит в состав дистрибутива или же бесплатно доступно для скачивания в Интернете. В этой главе мы рассмотрим три FTP-сервера: ftpd (стандарт в мире BSD), более продвинутый и самый популярный в мире Linux — ProFTPD и vsftp.

30.2. Настройка стандартного ftpd

30.2.1. Запуск ftpd и проверка работоспособности

Сервер ftpd устанавливается по умолчанию — именно поэтому он так популярен в FreeBSD — не приходится ничего доустанавливать, надо только обеспечить его автоматический запуск. Сервер может работать как в автономном (standalone) режиме,

так и запускаться через inetd. Для автономного запуска, когда ftpd будет загружаться каждый раз при запуске системы, нужно в файл /etc/rc.conf добавить строки:

```
ftpd_enable="YES"
ftpd_program="/usr/libexec/ftpd"
ftpd_flags=""
```

Первая запись обеспечивает автоматический запуск ftpd, вторая — устанавливает путь к исполнимому файлу сервера, а третья — задает параметры запуска ftpd. Теперь, чтобы не перезагружать компьютер, можно запустить ftpd командой:

```
# /etc/rc.d/ftpd start
```

Чтобы запускать ftpd через inetd по требованию (при наличии FTP-соединений), а не каждый раз при загрузке системы (если FTP-сервер используется редко, этот метод снижает нагрузку на систему), следует сначала активировать inetd. Для этого добавьте в файл /etc/rc.conf строку:

```
inetd_enable="YES"
```

Затем отредактируйте файл /etc/inetd.conf — раскомментируйте строку:

```
ftp stream tcp nowait root /usr/libexec/ftpd ftpd -l
```

После этого сохраните файл inetd.conf и запустите (или перезапустите, если он уже был запущен) inetd:

```
# /etc/rc.d/inetd start
```

или

```
# /etc/rc.d/inetd restart
```

Проверим доступность нашего сервера:

```
# ftp localhost
```

Вы должны увидеть ответ сервера (рис. 30.1).

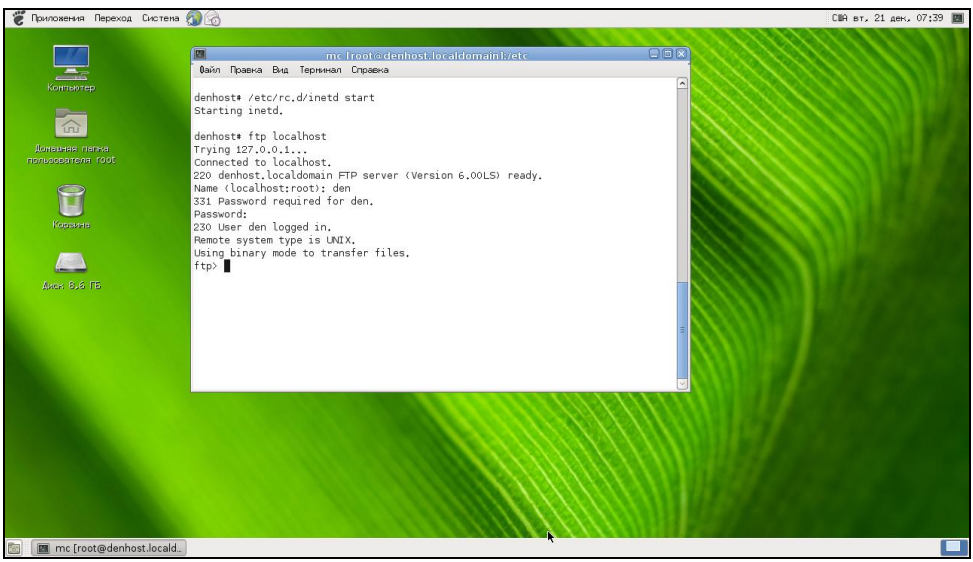


Рис. 30.1. Сервер запущен и принимает соединения

Теперь нужно перейти на другой компьютер сети и установить соединение с нашим FTP-сервером, чтобы убедиться, что он доступен другим компьютерам, а не только локальному.

30.2.2. Настройка сервера

При регистрации на сервере нам нужно указать имя пользователя и пароль. Сервер FTP получает информацию о пользователях из общесистемной базы — другими словами, создать FTP-пользователя можно все той же командой `adduser`, никакими иными средствами пользоваться не нужно.

Сервер ftpd позволяет настроить также анонимный доступ, когда вместо имени пользователя указывается логин `anonymous` (или `guest`), а вместо пароля — его e-mail. Анонимный доступ полезен, когда вам нужно распространять общедоступные файлы — не будете же вы для каждого пользователя создавать учетную запись?

А вот если FTP-сервер используется для хранения персональных файлов, тогда нужно создавать учетные записи пользователей. Можно настроить ftpd так, что он будет принимать регистрацию и обычных, и анонимных пользователей: при регистрации обычного пользователя FTP-клиент получит доступ к личным файлам пользователя, хранящимся на FTP-сервере, а при регистрации анонимного пользователя ему будет предоставлен доступ только к общему каталогу.

В файле /etc/ftpusers прописываются пользователи, которым запрещена регистрация по FTP, — по одному пользователю в каждой строке:

```
root
operator
toor
daemon
…
```

Вообще-то, когда пользователь указывает реальные имя и пароль, он получает доступ не только ко своему домашнему каталогу, а ко всей файловой системе. Понятно, что изменять файлы ему доступно только в своем каталоге, а вот прочитать он может практически любой файл, что не очень хорошо из соображений безопасности. Чтобы это предотвратить, в файле /etc/ftpchroot указываются пользователи, которые после регистрации в системе попадают в так называемую "песочницу" — в chroot-окружение, благодаря которому они не могут выйти за пределы своего домашнего каталога.

Итак, представим, что есть пользователи denis, max, igor. Пользователю denis вообще надо запретить FTP-доступ, поэтому его нужно добавить в файл /etc/ftpusers. Двум другим пользователям FTP-доступ нужен, но из соображений безопасности рекомендуется их добавить в файл /etc/ftpchroot:

```
max
igor
```

Когда пользователей много, можно объединить их в группу и указать имя группы с помощью символа `@`:

```
@ftp_users
```

В каталоге /etc вы можете отредактировать файлы ftpwelcome и ftpmotd. Первый содержит приветствие, отображаемое сразу после подключения пользователя к серверу, а второй — приветствие, выводимое после успешной регистрации на сервере.

В файле /var/ftp содержатся файлы, относящиеся к анонимному FTP-серверу. В частности, в каталоге /var/ftp/etc находится файл ftpmotd, содержимое которого выводится при регистрации анонимного пользователя.

Каталог /var/ftp/pub является публичным, в него нужно поместить общедоступные файлы. Каталог /var/ftp/incoming используется для загрузки файлов анонимными пользователями.

Настроить анонимный сервер просто — запустите sysinstall и перейдите по меню **Configure | Networking | Anon FTP**. Вы увидите вопрос о включении анонимного FTP-доступа — ответьте **Yes**, и откроется окно с параметрами анонимного доступа (рис. 30.2). Изменять здесь ничего не нужно, просто нажмите **OK.**

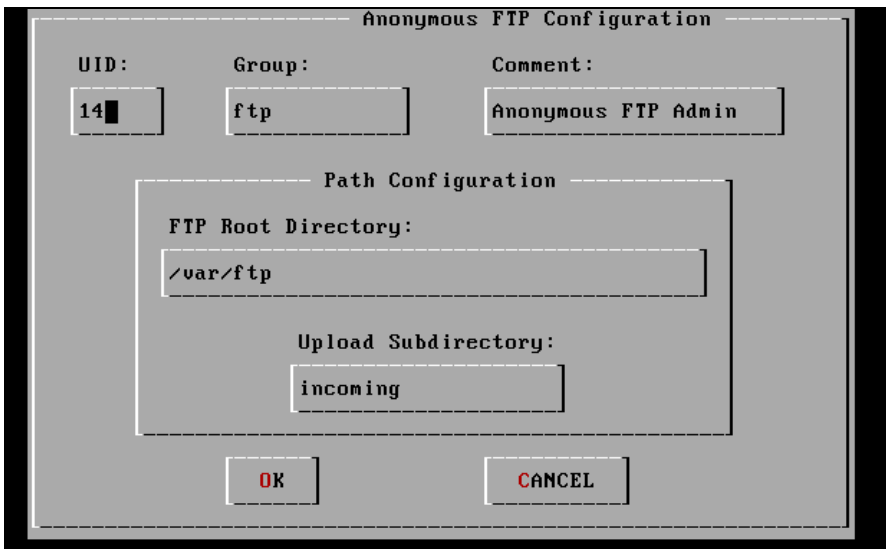


Рис. 30.2. Параметры анонимного FTP

Вам будет предложено изменить файл приветствия для анонимных пользователей (это можно будет сделать потом вручную с помощью любимого текстового редактора, поэтому сейчас лучше отказаться от редактирования этого файла). Далее вам сообщат, что для того чтобы запретить анонимным пользователям записывать файлы в /var/ftp/incoming, нужно запустить ftpd с параметром `-r`. Все бы хорошо, но тогда обычные пользователи ничего не смогут сохранить в своих каталогах, поэтому нас такой параметр не устраивает. Гораздо проще установить права, запрещающие запись в этот каталог:

```
# chmod 1555 /var/ftp/incoming
```

Чтобы потом вернуть все, как было, нужно установить права 1777:

```
# chmod 1777 /var/ftp/incoming
```

30.3. Сервер ProFTPD

30.3.1. Установка и запуск сервера

Как видно из названия, данный сервер является "профессиональной версией" сервера ftpd. Он значительно более гибок в настройке, и если ftpd можно использовать для построения тривиального (не в смысле tftp, а в смысле простого) FTP-сервера, то ProFTPD подходит для более серьезных конфигураций.

Установим ProFTPD из портов:

```
# cd /usr/ports/ftp/proftpd
# make install clean
```

Далее обеспечим автоматический запуск сервера, добавив следующую строку в файл /etc/rc.conf:

```
proftpd_enable="YES"
```

Для запуска, останова и перезапуска сервера будет служить команда:

```
# /usr/local/etc/rc.d/proftpd <start|stop|restart>
```

30.3.2. Конфигурационный файл сервера

Основным конфигурационным файлом сервера ProFTPD является файл /usr/local/etc/proftpd.conf. В листинге 30.1 представлен его простейший пример.

Листинг 30.1. Пример файла конфигурации /usr/local/etc/proftpd.conf

```
ServerName              "My server"          # можно написать все,
                                             # что угодно
ServerType              standalone           # автономный
DeferWelcome            off                  # вывести приветствие до
                                             # аутентификации
UseIPv6                 on                   # Можете установить off

DefaultServer           on                   # сервер по умолчанию
ShowSymlinks            on                   # показывать симв. ссылки

# настройка тайм-аутов
TimeoutNoTransfer       600
TimeoutStalled          600
TimeoutIdle             1200

# запрещает использовать данное выражение в FTP-командах
# (все файлы (маска *.*) вы уже не сможете удалить, придется удалять по одиночке!)
```

```
DenyFilter          \*.*/

Port                21                  # стандартный порт

MaxInstances        30                  # количество копий proftpd

# пользователь и группа, от имени которых работает proftpd
User                nobody
Group               nogroup

Umask               022                 # см. man umask

AllowOverwrite      on                  # можно перезаписывать файлы
```

В конфигурационном файле profptd.conf вы можете использовать обычные директивы, задающие одиночные свойства, и блочные директивы, определяющие группы свойств (параметров). Например, директива `ServerName` — обычная, она задает одно свойство, а директива `Directory` (см. далее) — блочная, позволяющая задать несколько параметров для одного каталога.

В табл. 30.1 приведены самые полезные директивы файла конфигурации profptd.conf. С остальными вы всегда можете ознакомиться, прочитав документацию по ProFTPD.

Таблица 30.1. *Директивы файла конфигурации profptd.conf*

Директива	Описание
`AccessGrantMsg "сообщение"`	Задает сообщение, которое будет отправлено пользователю при его регистрации на сервере. Можно задать грозное сообщение, напоминающее о том, что попытка несанкционированного доступа карается статьей такой-то уголовного кодекса
`Allow from all | узел | сеть [,узел | сеть[, ...]]`	Данная директива может использоваться только в блоке `Limit`. Директива разрешает доступ к серверу. По умолчанию используется значение `all`, которое разрешает доступ к серверу всем узлам со всех сетей
`AllowAll`	Разрешает доступ всем. Может использоваться в блоках `Directory`, `Anonymous`, `Limit`
`AllowForeignAddress on | off`	Разрешает узлу при подключении к серверу указывать адрес, не принадлежащий ему. По умолчанию используется значение `off` (то есть доступ запрещен), рекомендуется не изменять его. Директива может использоваться в блоках `Anonymous`, `<Global>`
`AllowGroup список групп`	Разрешает доступ к серверу указанным группам пользователей (группы должны быть зарегистрированы на этом сервере)

***Таблица 30.1** (продолжение)*

Директива	Описание
`AllowOverwrite on | off`	Разрешает (`on`) перезаписывать существующие файлы
`AllowUser список пользователей`	Разрешает доступ к серверу указанным группам пользователей (пользователи должны быть зарегистрированы на этом сервере)
`<Anonymous каталог>`	Разрешает анонимный доступ к указанному каталогу. Указанный каталог будет корневым каталогом анонимного FTP-сервера
`AuthGroupFile файл`	Задает альтернативный файл групп. По умолчанию /etc/group
`AuthUserFile файл`	Задает альтернативный файл паролей. По умолчанию /etc/passwd
`Bind IP-адрес`	Выполняет привязку дополнительного адреса к FTP-серверу
`DeferWelcome on | off`	Вывести приветствие после аутентификации (`on`) или до нее (`off`)
`Deny from all | узел | сеть`	Директива запрещает доступ к FTP-серверу. Используется в блоке `Limit`
`DenyAll`	Запрещает доступ всем к объектам, указанным в `Directory`, `Anonymous`, `Limit`
`DenyUser список пользователей`	Запрещает доступ указанным пользователям
`DefaultRoot каталог`	Определяет корневой каталог FTP-сервера. В качестве значения этого параметра полезно указать символ ~, тогда в качестве корневого каталога будет использоваться домашний каталог пользователя, который зашел на сервер
`DisplayLogin файл`	Указанный текстовый файл будет отображен, когда пользователь зайдет на сервер
`DisplayFirstChdir файл`	Отобразить указанный файл при каждой смене каталога
`<Directory каталог>`	Задает параметры доступа к каталогу и его подкаталогам
`<Global>`	Задает глобальные параметры FTP-сервера
`<Limit команда>`	Накладывает ограничение на выполнение FTP-команд, например, `READ`, `WRITE`, `STOR`, `LOGIN`
`MaxClients число сообщение`	Максимальное количество одновременно работающих клиентов. Если указанное число будет превышено, FTP-сервер отобразит указанное сообщение
`MaxLoginAttempts`	Максимальное количество попыток регистрации на сервере. По умолчанию 3. Указывается в блоке `Global`

Таблица 30.1 *(окончание)*

Директива	Описание
`MaxInstances`	Максимальное количество одновременно работающих экземпляров демона proftpd
`ServerType тип`	Задает тип запуска сервера. Значение по умолчанию — `standalone` (автономный запуск). Не нужно его изменять
`ServerName "имя"`	Задает имя сервера. Можете написать все, что угодно, например, `My server`
`ServerAdmin e-mail`	Позволяет указать адрес электронной почты администратора сервера
`ShowSymlinks on | off`	Показывать символические ссылки (`on`) или сразу результирующие файлы (`off`)
`Order allow, deny | deny, allow`	Задает порядок выполнения директив `Allow` и `Deny` в блоке `Limit`
`TimeoutIdle секунды`	Определяет тайм-аут простоя. Если пользователь не проявит активности за указанное время, соединение будет разорвано. По умолчанию используется значение 60
`TimeoutNoTransfer секунды`	Тайм-аут начала передачи. Сколько времени нужно ждать до разъединения, если пользователь вошел, но не начал передачу
`TimeoutStalled секунды`	"Замирание" во время передачи файла. Бывает такое, что клиент начал передачу (или прием) файла, но связь оборвалась. Этот тайм-аут определяет, сколько нужно ждать до разъединения в такой ситуации. Данный тайм-аут нужен, потому что бывает другая ситуация — когда у пользователя очень медленный канал
`Umask маска`	Задает права доступа для созданного файла
`User имя_пользователя`	Пользователь, от имени которого работает демон ProFTPD

30.3.3. Настройка реального сервера

Здесь мы настроим реальный FTP-сервер, к которому смогут получить доступ как обычные (зарегистрированные) пользователи, так и анонимные.

Приведенная в листинге 30.1 конфигурация вполне работоспособная и может использоваться для создания обычного (не анонимного) FTP-сервера. Но в конфигурационный файл нужно добавить две директивы:

```
DefaultRoot    ~
MaxClients     20 "Server is full!!!"
```

Первая директива делает корневым домашний каталог пользователя. При этом пользователь не может выйти за пределы своего домашнего каталога, и, следова-

тельно, не может навредить системе, если администратор неправильно установил права к каким-нибудь системным каталогам.

Вторая директива ограничивает число одновременно работающих клиентов во избежание перегрузки сервера.

Остальные параметры вы можете задать по своему усмотрению. Рассмотрим несколько примеров использования блоков `Directory` и `Login`:

```
<Directory upload>
       <Limit READ>
              DenyAll
       </Limit>
       <Limit WRITE>
              AllowAll
       </Limit>
</Directory>
```

Директива `Directory` здесь определяет две директивы `Limit` для каталога upload. Первая из них запрещает всем читать этот каталог, а вторая — разрешает всем записывать новые файлы в этот каталог. Каталог upload, таким образом, полностью оправдывает свое название — только для закачки файлов.

Рассмотрим еще один пример, запрещающий доступ к серверу всех узлов из подсети 192.168.1.0

```
<Limit LOGIN>
       DenyAll
       Deny from 192.168.1.
</Limit>
```

Если нужно, наоборот, разрешить доступ к серверу только пользователей из сети 192.168.1.0, надо использовать следующий блок `Limit`:

```
<Limit LOGIN>
Order deny, allow           # порядок действия deny-allow
       DenyAll              # запрещаем доступ всем
       Allow from 192.168.1. # разрешаем доступ только из сети 192.168.1.0
</Limit>
```

Теперь перейдем к анонимному доступу. Для его организации нужно добавить в файл конфигурации следующую директиву `Anonymous`:

```
<Anonymous ~ftp>

User                        ftp
Group                       nogroup

# Определяем псевдоним "anonymous" для пользователя "ftp"
# Клиенты смогут войти под обоими именами

UserAlias                   anonymous ftp

# Все файлы принадлежат пользователю ftp
```

```
DirFakeUser   on ftp
DirFakeGroup on ftp

# Не нужно требовать "правильную" оболочку
# "Правильной" считается оболочка, указанная в /etc/shells
RequireValidShell           off

# Максимальное число анонимных пользователей
MaxClients                  10

# Файлы с сообщениями
DisplayLogin                welcome.msg
DisplayFirstChdir           .message

# Ограничим WRITE для анонимных пользователей
   <Directory *>
     <Limit WRITE>
       DenyAll
     </Limit>
   </Directory>

</Anonymous>
```

30.4. Сервер vsftpd

30.4.1. Почему именно vsftpd?

Здесь мы рассмотрим настройку сервера vsftpd (Very Secure ftpd) — прогрессивного и компактного сервера с хранением виртуальных пользователей в базе MySQL и шифрованием паролей. Преимущество такого сервера заключается в том, что учетные записи хранятся в базе данных MySQL. А это означает, что в случае взлома одной из учетных записей ее нельзя будет использовать для входа в систему другим способом, например, по SSH. Также это полезно, если у FTP-сервера огромное количество пользователей, например, 10 000 — не нужно будет создавать в системе всех этих пользователей, достаточно добавить сколько угодно виртуальных пользователей в таблицу MySQL и не засорять системные файлы.

30.4.2. Установка сервера vsftpd и всего необходимого

Сам сервер vsftpd ничего не знает о существовании MySQL и виртуальных пользователей. Он использует модуль аутентификации (PAM) для получения информации о пользователей. Для нашей конфигурации достаточно установить нужный модуль (чтобы он брал информацию о пользователях из MySQL).

Установим сервер vsftpd и модуль `pam_mysql`:

```
# cd /usr/ports/ftp/vsftpd/
# make install clean

# cd /usr/ports/security/pam-mysql
# make install clean
```

Далее создаем ссылку:

```
# ln -sf /usr/local/lib/pam_mysql.so /usr/lib/pam_mysql.so
```

30.4.3. Создание базы данных MySQL

Будем считать, что сервер MySQL установлен и запущен (подробно настройка MySQL-сервера рассматривается в *главе 34*). Нужно создать базу данных и таблицу, где мы будем хранить учетные записи виртуальных пользователей. А также надо создать отдельного MySQL-пользователя для обращения к этой базе данных.

Сначала создадим базу данных командой:

```
# mysql -u root
```

Увидев командную строку mysql, введите следующий запрос (он создает базу данных `vsftpd` и предоставляет доступ к ней пользователю `vsftpd`):

```
CREATE DATABASE vsftpd;
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, \
DROP ON vsftpd.* TO 'vsftpd'@'localhost' IDENTIFIED BY 'ftpdpass';
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, \
DROP ON vsftpd.* TO 'vsftpd'@'localhost.localdomain' IDENTIFIED BY 'ftpdpass';
FLUSH PRIVILEGES;
```

Создадим таблицу, содержащую данные о виртуальных учетных записях:

```
USE vsftpd;
CREATE TABLE `accounts` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`username` VARCHAR( 30 ) NOT NULL ,
`pass` VARCHAR( 50 ) NOT NULL ,
UNIQUE (
`username`
)
) ENGINE = MYISAM ;
INSERT INTO accounts (username, pass) VALUES('denis', PASSWORD('123456'));
```

Первый запрос выбирает базу данных (`vsftpd`), второй — создает таблицу `accounts`, хранящую информацию о виртуальных пользователях. Третий — добавляет виртуального пользователя `denis` с паролем `123456`.

Для выхода из mysql введите команду:

```
quit;
```

Теперь добавим пользователя `vsftpd`:

```
# pw useradd vsftpd -c "virtual" -s /sbin/nologin -d /home/vsftpd
```

Примечание

Если вам удобно использовать команду `adduser` — пожалуйста, только укажите в ней домашний каталог `/home/vsftpd` и оболочку `nologin`.

30.4.4. Конфигурационный файл сервера vsftpd

Основным файлом конфигурации сервера vsftpd является файл /usr/local/etc/vsftpd.conf, представленный в листинге 30.2.

Листинг 30.2. Файл /usr/local/etc/vsftpd.conf

```
anonymous_enable=NO          # нет анонимного доступа
local_enable=YES
write_enable=YES             # разрешена запись
local_umask=022
anon_upload_enable=NO        # анонимная запись запрещена
anon_mkdir_write_enable=NO
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
xferlog_file=/var/log/vsftpd.log
nopriv_user=vsftpd

chroot_local_user=YES
secure_chroot_dir=/usr/local/share/vsftpd/empty
listen=YES
# Диапазоны пассивных портов, эти диапазоны нужно "прописать" в
# настройках брандмауэра
pasv_min_port=30000
pasv_max_port=30100

pam_service_name=vsftpd
guest_enable=YES
guest_username=vsftpd
# Задает каталог для виртуального пользователя — это подкаталог
# каталога /home/vsftpd
local_root=/home/vsftpd/$USER
user_sub_token=$USER
virtual_use_local_privs=YES
user_config_dir=/usr/local/etc/vsftpd_user_conf
```

Можете оставить этот файл как есть. Мне не нужен анонимный доступ, поэтому первую директиву я установил в `NO`. Самая важная строка выделена полужирным

шрифтом и подчеркнута — она заставляет сервер использовать PAM[1]-сервер vsftpd. Но этот сервис еще нужно создать:

ee /etc/pam.d/vsftpd

В файл /etc/pam.d/vsftpd добавьте две строки:

```
auth required pam_mysql.so user=vsftpd passwd=ftpdpass
host=localhost db=vsftpd table=accounts usercolumn=username passwdcolumn=pass
crypt=2
account required pam_mysql.so user=vsftpd passwd=ftpdpass
host=localhost db=vsftpd table=accounts usercolumn=username passwdcolumn=pass
crypt=2
```

ПРИМЕЧАНИЕ

Эти строки столь длинные, что их нельзя поместить без разрывов на страницу книги, поэтому проследите, чтобы первая строка начиналась с `auth required`, а вторая — с `account required`. В строках не должно быть переносов, а между строками — пробелов.

Нужно также создать каталог, упомянутый в конфигурационном файле:

```
# mkdir /usr/local/etc/vsftpd_user_conf
```

Чуть раньше мы с помощью SQL-запроса создали виртуального пользователя `denis`, теперь надо создать для него домашний каталог:

```
mkdir /home/vsftpd/denis
chown vsftpd:nogroup /home/vsftpd/denis
```

В файл /etc/rc.conf добавьте строку для автоматического запуска vsftpd:

```
vsftpd_enable="YES"
```

Вот, собственно, и все — перезагружаемся и проверяем FTP-сервер:

```
# ftp localhost
```

Connected to 192.168.181.6.
220 Welcome to denhost FTP service.
Name (192.168.181.6:root): `denis`
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>

[1] pam-mysql — популярный встраиваемый модуль аутентификации (PAM, Pluggable Authentication Modules, подгружаемые модули аутентификации), который позволяет производить аутентификацию с использованием MySQL.

Глава 31



NFS — сетевая файловая система

31.1. Принцип работы NFS

BSD поддерживает много разных файловых систем, в том числе и сетевую файловую систему (NFS, Network File System). Сетевая файловая система позволяет другим машинам использовать экспортируемую часть файловой системы нашей машины. Точно так же, наша машина может использовать экспортированные части файловой системы других машин (по сути, получается аналогия с механизмом общего доступа к файлам и папкам в Windows).

Использование NFS дает следующие преимущества:

- экономию дискового пространства — общие файлы, которые могут понадобиться всем пользователям, можно хранить на одной центральной машине, а все рабочие станции подмонтируют экспортируемый каталог и будут использовать эти файлы так, как если бы они находились на собственном, локальном диске;
- хранение домашних каталогов на центральной машине — домашние каталоги пользователей можно хранить на центральной машине, что сделает их доступными при регистрации с любой машины сети. Вам это ничего не напоминает из мира Windows?
- уменьшение количества устройств для чтения/записи сменных носителей — можно экспортировать приводы DVD, Zip, кардридеры и использовать их удаленно. Соответственно, понадобится меньшее количество этих самых приводов, что тоже обеспечит определенную экономию. Конечно, будет несколько неудобно, но экономия — налицо.

NFS состоит из двух частей: сервера и клиента. Клиент обращается к данным, экспортируемым сервером. На сервере должны быть запущены следующие демоны:

- nfsd — обслуживает запросы, поступающие от NFS-клиентов;
- mountd — демон монтирования, выполняющий запросы, которые ему передает nfsd;
- rpcbind — позволяет NFS-клиентам определить порт, используемый сервером NFS.

На клиенте нужно запустить только один демон — nfsiod, обслуживающий запросы, поступаемые от сервера NFS. Вообще-то его запуск необязателен, и для нормальной работы NFS он не нужен, но позволяет повысить производительность NFS.

31.2. Настройка и использование NFS

Настройка NFS — довольно простой процесс. Достаточно отредактировать всего два файла: /etc/rc.conf, в котором следует указать, какие демоны нужно запустить, и /etc/exports, в котором указываются экспортируемые файловые системы. Начнем с редактирования файла /etc/rc.conf.

Для настройки NFS на сервере в файл /etc/rc.conf надо добавить следующие строки:

```
rpcbind_enable="YES"
nfs_server_enable="YES"
nfs_server_flags="-u -t -n 4"
mountd_flags="-r"
```

Демон mountd отдельно запускать не нужно, поскольку он запустится автоматически при запуске nfsd.

На NFS-клиенте в файл /etc/rc.conf достаточно добавить всего одну строку:

```
nfs_client_enable="YES"
```

Вот теперь можно приступить к редактированию файла /etc/exports, который определяет, какие файловые системы будут экспортироваться (переданы в совместное использование).

В каждой строке файла /etc/exports указывается, какая файловая система будет экспортироваться и какие машины имеют право использовать эту файловую систему. Могут также указываться и параметры, влияющие на доступ к экспортируемой файловой системе.

Структура записей файла /etc/exports следующая:

```
<каталог> хост1(опции) хост2(опции)…хостN(опции)
```

Предположим, что мы хотим экспортировать DVD-привод. DVD-привод обычно монтируется к каталогу /cdrom, соответственно его и нужно экспортировать:

```
/cdrom -ro    comp1 comp2
```

Этой строкой доступ к DVD предоставляется только компьютерам `comp1` и `comp2`, а параметр `-ro` означает "только чтение", то есть запрет изменения файлов экспортируемой файловой системы:

Вместо имен компьютеров можно указать IP-адреса:

```
/cdrom -ro    192.168.1.11 192.168.1.12
```

Вместо списка узлов можно указать символ `*` или IP-адрес сети, например:

```
/directory1   -ro    *
/directory2   -ro    192.168.1.0/28
```

Здесь первая запись экспортирует всем узлам каталог /directory1 в режиме "только чтение". Вторая запись экспортирует каталог /directory2 в режиме "только чтение" только первым 16 адресам: от 192.168.1.0 до 192.168.1.15. Узлу с адресом 192.168.1.16 и выше этот каталог будет недоступен.

Более того, можно вообще не указывать имена узлов. Например:

```
/cdrom (ro)
/pub   (ro,insecure,all_squash)
```

В первом случае каталог /cdrom станет доступным всем компьютерам сети в режиме "только чтение". Во втором случае каталог /pub экспортируется с рядом дополнительных параметров (табл. 31.1), но в общих чертах он будет доступен всем компьютерам в режиме "только чтение".

Таблица 31.1. *Некоторые опции экспортирования файловых систем*

Опция	Описание
`ro`	Разрешает только чтение
`rw`	Разрешает и чтение, и запись
`secure`	Запросы должны исходить из портов безопасного диапазона, то есть номер порта должен быть меньше 1024
`insecure`	Позволяет использовать любой номер порта
`root_squash`	Используется для преобразования запросов от пользователя root (uid=0) в запросы от анонимного пользователя. Благодаря этому пользователь root не будет обладать своими правами при доступе к экспортированной файловой системе
`no_root_squash`	Отменяет опцию `root_squash`. То есть пользователь root может пользоваться своими правами при доступе к экспортируемой файловой системе из NFS-клиента. Небезопасная опция!
`all_squash`	Все идентификаторы пользователей и групп будут конвертированы в анонимные. Самая безопасная опция
`no_all_squash`	Обратна опции `all_squash`, используется по умолчанию
`anonuid=UID` `anonguid=GID`	Задают идентификаторы анонимного пользователя и группы соответственно
`link_absolute`	Оставляет все символические ссылки без изменений. Используется по умолчанию
`link_relative`	Конвертирует все абсолютные ссылки в относительные

Если нужно экспортировать каталоги /directory1 и /directory2 только одному клиенту, то нужно указать их в одной записи так:

```
/directory1 /directory2      client
```

Когда мы экспортируем каталог со множеством подкаталогов, целесообразно разрешить клиенту монтировать только те каталоги, которые ему нужны. Предположим, что мы экспортируем каталог /home. С узла 192.168.1.11 некий пользователь pupkin захочет подмонтировать только подкаталог /home/pupkin — ему незачем монтировать весь каталог /home. Для этого ему нужно использовать опцию: `-alldirs`:

```
/home  -alldirs  192.168.1.11 192.168.1.12
```

Если нужно разрешить клиенту запись на экспортируемую файловую систему с правами root, то указывается параметр `-maproot`:

```
/opt  -maproot=root  192.168.1.11
```

Рассмотрим две записи:

```
/home  den.firma.com(rw)
/home  den.firma.com (rw)
```

Эти записи, несмотря на свою схожесть, выполняют совершенно разные действия. В первом случае каталог /home экспортируется только для компьютера den.firma.com, который может его смонтировать в режиме "чтение/запись" (параметр `rw`).

Во втором случае тоже экспортируется каталог /home, но клиенту den.firma.com он доступен в режиме "только чтение" (используется по умолчанию), а всем остальным компьютерам сети — в режиме "чтение/запись". А виной всему пробел между именем узла и скобкой с параметрами.

После изменения файла /etc/exports (чтобы изменения вступили в силу) нужно перезапустить демон mountd:

```
# /etc/rc.d/mountd reload
```

31.3. Монтирование экспортированной файловой системы на клиенте

Подмонтировать сетевую файловую систему на клиенте можно все той же командой `mount`. Рассмотрим пример монтирования файловой системы /home, находящейся на сервере, к каталогу /nfs на клиенте:

```
# mount server:/home /nfs
```

Если все настроено правильно, через каталог /nfs вы сможете обратиться к файлам, находящимся на сервере. Чтобы не вводить команду `mount` каждый раз при перезагрузке компьютере, целесообразно добавить соответствующую запись в файл /etc/fstab для автоматического монтирования сетевой файловой системы:

```
server:/home     /nfs    nfs     rw      0   0
```

Здесь `server:/home` — имя сервера и имя экспортируемой сервером файловой системы, `/nfs` — точка монтирования (этот каталог должен существовать), `nfs` — тип файловой системы, `rw` — доступ в режиме "чтение/запись". Последние два поля установлены в `0` для отключения проверки и дампа сетевой файловой системы.



Глава 32

Почтовый сервер

32.1. Выбор программного обеспечения

Для настройки почтового сервера нам понадобятся две программы: MTA (Mail Transfer Agent, агент передачи почты) и IMAP/POP3-сервер:

- MTA выполняет функцию сервера SMTP (Simple Mail Transfer Protocol, простой протокол передачи почты), то есть отправляет почту;
- сервер IMAP/POP3 позволяет удаленным пользователям подключиться к почтовику с помощью почтовой программы (The Bat!, Windows Live, Outlook Express и т. п.) и получить/просмотреть письма, находящиеся в их почтовых ящиках.

> ***Примечание***
>
> В качестве MTA на многих системах установлена программа sendmail, ставшая классикой. Однако в этой книге настройка sendmail рассматриваться не будет, и тому есть ряд причин. Во-первых, настройка sendmail описана во многих статьях, которые можно бесплатно найти и прочитать в Интернете. Да и в "хэндбуке" по FreeBSD эта программа достаточно подробно описана: **http://www.freebsd.org/doc/ru_RU.KOI8-R/books/handbook/sendmail.html**. Но, покупая эту книгу, вы платили деньги не за то же, что можно и так найти в Интернете? Поэтому я должен предложить вам более интересное решение, чем обычный sendmail. Во-вторых, sendmail весьма неудобна в настройке. Как подумаю, что придется опять редактировать ее конфигурационный файл, так сразу хочется снести sendmail и установить другой MTA с более читабельным и понятным конфигурационным файлом.

Так уж получилось, что вторым MTA, с которым я познакомился в далеком 2001 году, стала программа postfix. Postfix — довольно молодая программа, особенно на фоне sendmail. Еще бы — первая версия postfix появилась в середине 1999 года, а начало эксплуатации sendmail датируется 1979 годом. То есть на момент написания этих строк первой программе — чуть больше 12 лет, а второй — уже 32 года. Когда разрабатывался sendmail, на информационную безопасность никто не обращал особого внимания — компьютеры тогда были непозволительной роскошью. Совершенно закономерно, что при таком отношении к безопасности по мере дальнейшего развития проекта в sendmail "всплыли" дыры. В современных версиях sendmail замеченные недостатки устранены, но неприятный осадок остался. Помню, мой сайт **dhsilabs.narod.ru** как-то раз закрыли, потому что на нем было выложено руководство… по взлому sendmail.

Программа postfix изначально разрабатывалась как альтернатива sendmail. Считается, что postfix быстрее, надежнее и безопаснее своего основного конкурента. К тому же postfix благодаря модульной структуре расходует меньше системных ресурсов, особенно во время простоя почтовой системы, тогда как sendmail потребляет драгоценное процессорное время и оперативную память, даже если ничем не занят. Так что, можно сказать, что с выбором MTA мы определились — это будет postfix.

> ***Примечание***
>
> В других моих книгах рассматриваются иные программы — например, в "Серверном применении Linux"[1] описана QMail. А если уж вам захотелось настоящей эксклюзивности, можете попробовать программу Exim. Процесс настройки этой программы и POP3-сервера Dovecot на базе FreeBSD подробно изложен на следующей страничке: **http://sysadminpages.com/2009/07/freebsd-exim-dovecot-mysql/**.

А вот в качестве IMAP/POP3-сервера мы рассмотрим классику — Courier-IMAP. С этой программой я работаю довольно давно, и она не вызывает у меня каких-либо нареканий. Альтернативой ей может служить уже упомянутый ранее IMAP/POP3-сервер Dovecot — довольно удачный сервер с огромным (сразу вспомнил о sendmail) конфигурационным файлом, чем он меня как раз и отпугнул.

Кроме этих двух программ нам также понадобятся:

- сервер MySQL — для аутентификации средствами MySQL. Можно обойтись и без него, но с сервером баз данных наша конфигурация становится интереснее, к тому же при большой нагрузке использование MySQL повышает производительность системы;
- библиотека Cyrus-sasl2 — необходима для SASL-аутентификации при отправке почты (SMTP-аутентификация);
- Web-интерфейс PostfixAdmin — интерфейс для удаленного и простого управления postfix. Его можно тоже не использовать, но тогда все служебные таблицы вам придется править программой phpMyAdmin или любой другой, что не очень удобно;
- Web-сервер Apache с поддержкой PHP — нужен для запуска PostfixAdmin.

32.2. Установка вспомогательного ПО

Все программное обеспечение следует устанавливать из портов, поскольку только в этом случае можно включить необходимые нам опции (при установке программ из пакетов выбор опций не представляется возможным).

Будем считать, что Web-сервер с поддержкой PHP уже установлен. Подробно настройка Web-сервера с поддержкой PHP и MySQL рассмотрена в *главе 34*.

---

[1] Колисниченко Д. Н. Серверное применение Linux: 2-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2009, **http://www.bhv.ru/books/book.php?id=184941**.

32.2.1. Установка MySQL-сервера

MySQL-сервер должен быть собран с опцией `WITH_OPENSSL` (по умолчанию так оно и происходит). Но если вы еще не устанавливали MySQL, чтобы подстраховаться, установите переменную окружения `WITH_OPENSSL` перед сборкой программы (рис. 32.1):

```
# cd /usr/ports/databases/mysql51-server/
# setenv WITH_OPENSSL yes
# make install clean
# rehash
```

Обратите внимание, приведенные команды устанавливают MySQL-сервер версии 5.1. При желании вы можете установить версию сервера 5.5. Также желательно установить вспомогательные скрипты MySQL:

```
# cd /usr/ports/databases/mysql51-scripts/
# make install clean
# rehash
```

После установки MySQL запустите его и скрипт настройки:

```
# /usr/local/etc/rc.d/mysql-server start
# /usr/local/bin/mysql_secure_installation
```

Убедитесь, что MySQL-сервер и Web-сервер запускаются автоматически — в файле /etc/rc.conf должны присутствовать строки:

```
apache22_enable="YES"
mysql_enable="YES"
```



Рис. 32.1. Сборка MySQL с поддержкой OpenSSL

32.2.2. Установка библиотеки Cyrus-sasl2

Перед установкой библиотеки Cyrus-sasl2 следует установить OpenSSL, необходимый для создания ключей и сертификатов:

```
# cd /usr/ports/security/openssl
# make install clean
```

Теперь установим Cyrus-sasl2:

```
# cd /usr/ports/security/cyrus-sasl2
# make install clean
```

Перед сборкой нужно включить опции `MYSQL`, `LOGIN`, `PLAIN`, `CRAM`, `DIGEST` (рис. 32.2).

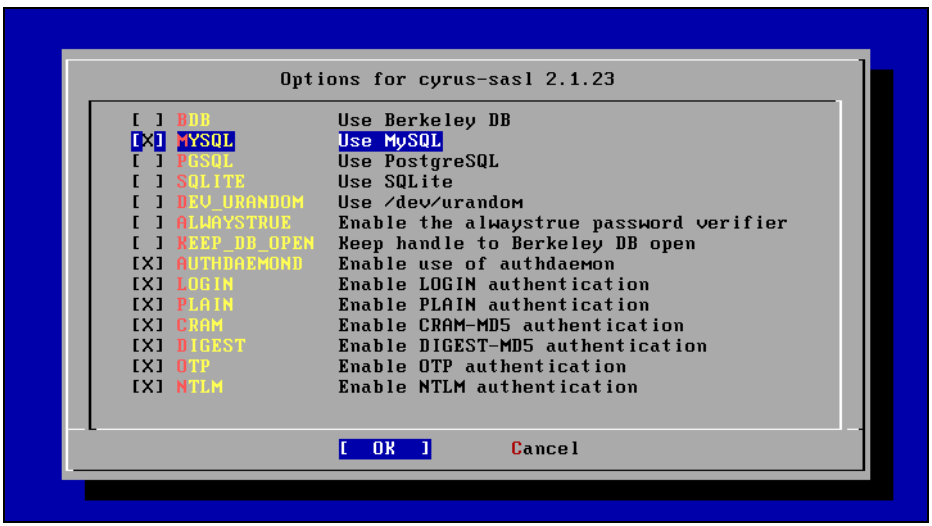


Рис. 32.2. Сборка библиотеки Cyrus-sasl2

32.2.3. Установка библиотеки Courier-authlib

Для аутентификации пользователей Courier-IMAP использует библиотеку Courier-authlib, установим ее:

```
# cd /usr/ports/security/courier-authlib/
# make config install clean
```

При сборке этой библиотеки нужно включить опцию `AUTH_MYSQL`, поскольку аутентификация будет производиться именно с помощью MySQL (рис. 32.3).

Сразу после установки Courier-authib добавьте следующую строку в файл /etc/rc.conf:

```
courier_authdaemond_enable="YES"'
```

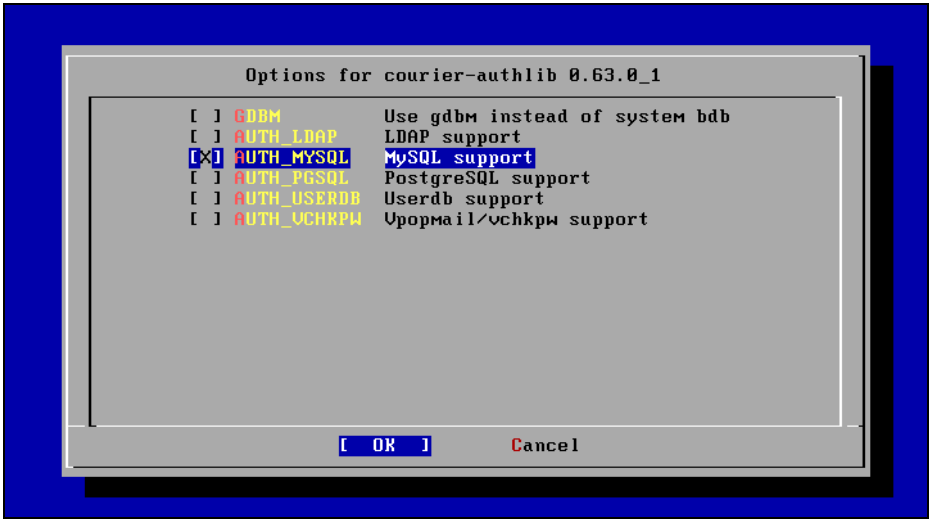


Рис. 32.3. Сборка Courier-authlib

32.2.4. Редактирование конфигурационных файлов

Теперь нужно отредактировать конфигурационные файлы /usr/local/etc/authlib/authdaemonrc и /usr/local/etc/authlib/authmysqlrc. Полностью листинги этих файлов я приводить не стану, скажу только, на какие опции нужно обратить внимание.

Откройте файл /usr/local/etc/authlib/authdaemonrc, изменять в нем ничего не нужно, просто найдите и установите опции, как показано в листинге 32.1.

Листинг 32.1. Фрагмент файла /usr/local/etc/authlib/authdaemonrc

```
authmodulelist="authmysql"
authmodulelistorig="authmysql"
daemons=5
authdaemonvar=/var/run/authdaemond
subsystem=mail
DEBUG_LOGIN=2
DEFAULTOPTIONS="wbnodsn=1"
LOGGEROPTS=""
```

Файл /usr/local/etc/authlib/authmysqlrc, если удалить все комментарии, должен выглядеть так, как показано в листинге 32.2.

Листинг 32.2. Файл /usr/local/etc/authlib/authmysqlrc без комментариев

```
MYSQL_USERNAME          postfix
MYSQL_PASSWORD          postfix
```

```
MYSQL_SOCKET            /tmp/mysql.sock
MYSQL_OPT               0
MYSQL_DATABASE          postfix
MYSQL_CHARACTER_SET     utf8
MYSQL_USER_TABLE        mailbox
MYSQL_CLEAR_PWFIELD     password
MYSQL_UID_FIELD         '5555'
MYSQL_GID_FIELD         '5555'
MYSQL_LOGIN_FIELD       username
MYSQL_HOME_FIELD        '/var/spool/mail/virtual'
MYSQL_NAME_FIELD        name
MYSQL_MAILDIR_FIELD     maildir
MYSQL_QUOTA_FIELD       quota
MYSQL_WHERE_CLAUSE      active='1'
```

Как вы уже успели догадаться, здесь указываются параметры доступа к серверу MySQL. Мы используем имя пользователя и пароль `postfix`, база данных названа так же.

Параметры `MYSQL_UID_FIELD` и `MYSQL_GID_FIELD` задают идентификаторы пользователя и группы `virtual`, которые будут созданы чуть позже. Идентификатор мы указали заранее, будем надеяться, что в вашей системе нет пользователя и группы с идентификатором `5555`.

Если сервер MySQL находится на другой машине, то вместо параметра `MYSQL_SOCKET` нужно использовать параметры, задающие имя сервера и номер порта, например:

```
MYSQL_SERVER  localhost
MYSQL_PORT    3306
```

Запустим authdaemond:

```
# /usr/local/etc/rc.d/courier-authdaemond start
```

Starting courier_authdaemond.

Вся подготовительная работа сделана. Теперь можно приступить к установке IMAP/POP3-сервера.

32.3. Установка Courier-IMAP

Настало время установить IMAP/POP3-сервер Courier-IMAP. Установку postfix пока отложим — там есть с чем повозиться. А вот в установке Courier-IMAP нет ничего сложного:

```
# cd /usr/ports/mail/courier-imap/
# make config install clean
```

Поддержку IPv6 можно отключить, но обязательно включите опции `TRASHQUOTA` и `AUTH_MYSQL` (рис. 32.4).

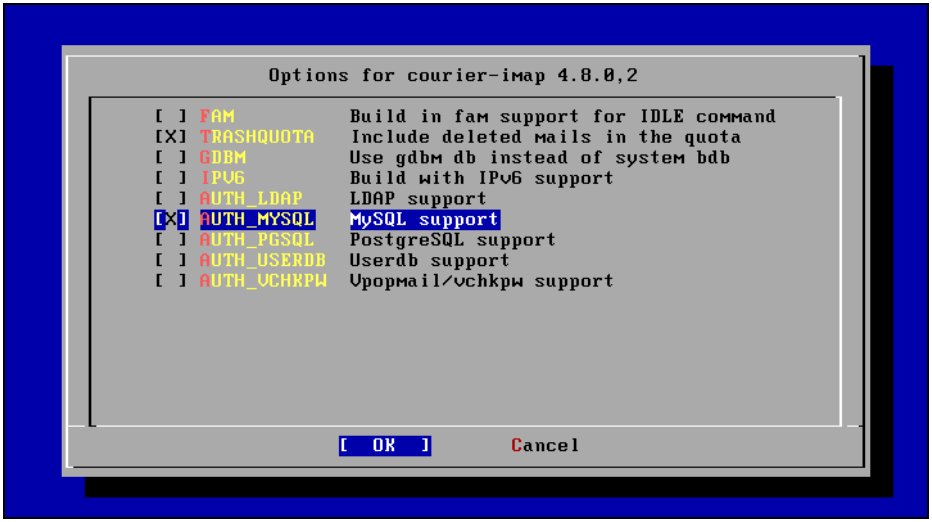


Рис. 32.4. Сборка Courier-IMAP

После установки сервера следует отредактировать его конфигурационный файл /usr/local/etc/courier-imap/pop3d. В листинге 32.3 представлен этот файл без комментариев (чтобы уменьшить длину листинга).

Листинг 32.3. Файл /usr/local/etc/courier-imap/pop3d

```
PIDFILE=/var/run/pop3d.pid
MAXDAEMONS=40
MAXPERIP=4
POP3AUTH="PLAIN LOGIN CRAM-MD5"
POP3AUTH_ORIG="PLAIN LOGIN CRAM-MD5"
POP3AUTH_TLS="PLAIN LOGIN CRAM-MD5"
POP3AUTH_TLS_ORIG="PLAIN LOGIN CRAM-MD5"
POP3_PROXY=0
PORT=110
ADDRESS=192.168.1.1
TCPDOPTS="-nodnslookup -noidentlookup"
LOGGEROPTS="-name=courier-imap"
POP3DSTART=YES
MAILDIRPATH=Maildir
```

Большинство параметров можно оставить по умолчанию, но назначение некоторых вам обязательно надо знать — не зря же был приведен этот листинг. Параметр `POP3_PROXY` задает адрес прокси, если такой есть в вашей сети. Параметр `PORT` содержит номер порта для POP3 — обычно это `110`.

Параметр ADDRESS задает адрес интерфейса, который надо прослушивать в ожидании соединения с клиентами (почтовыми программами пользователей). Значение 0 означает, что нужно прослушивать все интерфейсы. Если интерфейсов много, а прослушать нужно не все, а некоторые (то есть число прослушиваемых интерфейсов больше 1, но меньше общего числа интерфейсов), следует указать их явно в опции PORT (просто укажите IP-адреса интерфейсов, добавив к ним номер порта):

```
ADDRESS=0
PORT=192.168.1.1.110, 192.168.2.1.110, 127.0.0.1.110
```

Если все готово, запускаем IMAP/POP3-сервер:

```
# /usr/local/etc/rc.d/courier-imap-pop3d start
```

Starting courier_imap_pop3d.

Проверим, запустился ли он:

```
# sockstat | grep :110
```

root couriertcp 7362 3 tcp4 192.168.1.1:110 *:*

Сервер запущен, его PID — 7362. Не забудьте обеспечить его автоматический запуск, добавив следующую строку файл в /etc/rc.conf:

```
courier_imap_pop3d_enable="YES"
```

32.4. Установка postfix

Перед установкой postfix отключите sendmail, если эта программа установлена в вашей системе, чтобы два MTA не конфликтовали между собой. Для этого найдите в файле rc.conf все, что связано с sendmail:

```
# cat /etc/rc.conf | grep sendmail
```

Или удалите все найденные строки, или установите для них значение NO:

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
```

В файле /etc/periodic.conf отключите следующие опции:

```
daily_clean_hoststat_enable="NO"
daily_status_mail_rejects_enable="NO"
daily_status_include_submit_mailq="NO"
daily_submit_queuerun="NO"
```

Вот теперь можно смело приступать к установке postfix:

```
# cd /usr/ports/mail/postfix
# make config install clean
```

Установите опции postfix, как показано на рис. 32.5. Мы включаем поддержку Perl-совместимых регулярных выражений, SASL, VDA (Virtual Delivery Agent, виртуальный агент доставки), TLS и MySQL.

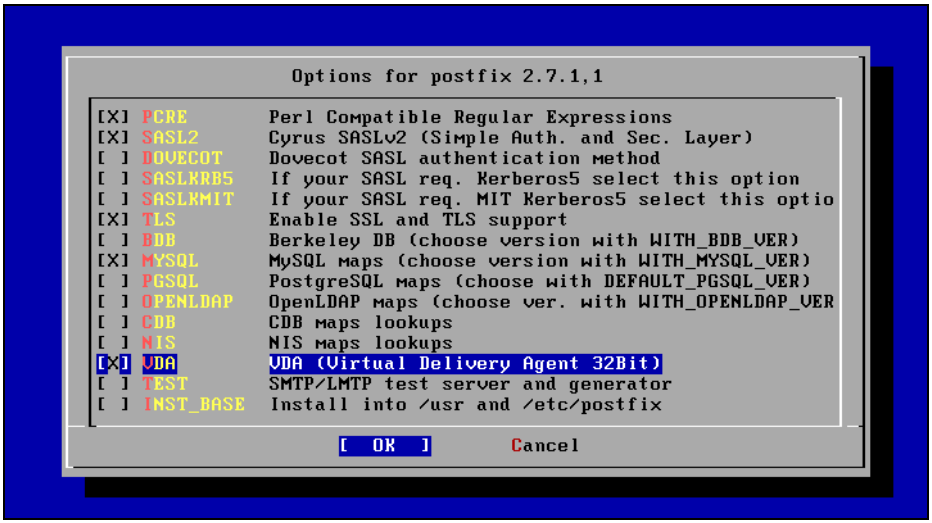


Рис. 32.5. Сборка postfix

При установке postfix нужно обязательно ответить `y` на следующие вопросы:

You need user «postfix» added to group «mail».

Would you like me to add it [y]? `y`

Would you like to activate Postfix in /etc/mail/mailer.conf [n]? `y`

Таким образом мы добавили пользователя postfix в группу mail, а также прописали postfix в файл /etc/mail/mailer.conf.

Сразу после установки обеспечим автозапуск postfix (чтобы потом не забыть), добавив вот эту строку в файл /etc/rc.conf:

```
postfix_enable="YES"
```

Отредактируем файл конфигурации /usr/local/etc/postfix/main.cf. По сравнению с конфигурационным файлом sendmail, файл main.cf не очень большой, но для экономии места в книге полностью мы его рассматривать не будем, а сконцентрируемся только на самых важных параметрах, прокомментированных в листинге 32.4.

Листинг 32.4. Самые важные параметры postfix (файл main.cf)

```
# Владелец процесса, запускать postfix от имени root небезопасно
mail_owner = postfix

# имя почтового сервера
myhostname = mail.dkws.org

# Наш домен
mydomain = dkws.org
```

```
# Слушать все сетевые интерфейсы
inet_interfaces = all

# Список доменов, для которых почта доставляется локально
# Не указывайте здесь виртуальные домены — для этого есть
# параметр virtual_mailbox_domains
mydestination = $myhostname, localhost.$mydomain, localhost

# Наши сети, этим сетям разрешено пересылать почту через postfix
# Можно указать всех клиентов нашей сети, а можно указать просто
# 127.0.0.0/8 — сервер сможет принимать почту, но пересылка
# почты будет запрещена
mynetworks = 127.0.0.0/8
#mynetworks = 192.168.1.0/24 127.0.0.0/8

# Отклонять команду ETRN
smtpd_etrn_restrictions = reject

# Отклоняем почту с неизвестным отправителем
# Позволяет уберечься от червей и вирусов
smtpd_reject_unlisted_sender = yes

# Отключаем команду VRFY, чем запрещаем проверку существования
# почтового ящика — сделаем спаммерам дополнительную преграду
disable_vrfy_command = yes

# Адреса в командах MAIL FROM и RCPT TO должны заключаться в <>
strict_rfc821_envelopes = yes

# Требуем, чтобы удаленный клиент представился командой HELO или EHLO
smtpd_helo_required = yes

# Число ошибок SMTP-клиента, превышение этого лимита ведет к разрыву
# соединения
smtpd_hard_error_limit = 5

# Поддержка SASL-аутентификации
smtpd_sasl_auth_enable = yes

# Включаем поддержку старых почтовых клиентов вроде outlook express 4
# Они используют старую версию команды AUTH, описанную в RFC 2554
broken_sasl_auth_clients = yes

# Запрещаем анонимную аутентификацию
```

```
smtpd_sasl_security_options = noanonymous

# Информация об отправителях
smtpd_sender_login_maps = mysql:$base/mysqlLookupMaps/sender.conf

# Отсюда postfix должен брать информацию о псевдонимах
virtual_alias_maps = mysql:$base/mysqlLookupMaps/alias.conf

# Отсюда postfix должен брать информацию о доменах
virtual_mailbox_domains = mysql:$base/mysqlLookupMaps/domain.conf

# Карты почтовых ящиков и сами почтовые ящики. Обратите внимание на
# имя каталога virtual_mailbox_base — оно отличается от имени по
# умолчанию (/var/spool/mail). Это каталог для виртуальных ящиков
virtual_mailbox_maps = mysql:$base/mysqlLookupMaps/mailbox.conf
virtual_mailbox_base = /var/spool/mail/virtual

# Поддержка квот
virtual_mailbox_limit_maps = mysql:$base/mysqlLookupMaps/quota.conf
virtual_maildir_extended=yes
virtual_mailbox_limit_override=yes
virtual_create_maildirsize = yes
virtual_overquota_bounce = yes
virtual_maildir_limit_message="Ups, mailbox is full"

# Максимальный размер письма 10 Мбайт
message_size_limit = 10485760

# 5555 — uid и gid пользователя и группы virtual
virtual_gid_maps = static:5555
virtual_uid_maps = static:5555
```

Теперь следует указать postfix, как нужно производить аутентификацию. Для этого надо отредактировать файл /usr/local/lib/sasl2/smtpd.conf, как показано в листинге 32.5.

Листинг 32.5. Файл /usr/local/lib/sasl2/smtpd.conf

```
pwcheck_method: auxprop
log_level: 3
mech_list: PLAIN LOGIN CRAM-MD5
auxprop_plugin: sql
sql_usessl: yes
sql_engine: mysql
```

```
sql_hostnames: localhost
sql_user: postfix
sql_passwd: postfix
sql_database: postfix
sql_select: select password from mailbox where username = '%u@%r'
```

Поскольку мы используем MySQL-аутентификацию, необходимо указать параметры доступа к MySQL. Мы используем ранее указанные параметры: сервер — `localhost`, имя пользователя, пароль и база данных — `postfix`.

Чуть не забыл. Надо инициализировать базу данных псевдонимов:

```
# /usr/local/bin/newaliases
```

Теперь нужно создать каталог mysqlLookupMaps, который так часто упоминался в конфигурационном файле main.cf

```
# mkdir /usr/local/etc/postfix/mysqlLookupMaps
```

Создаем все необходимые файлы (они тоже указаны в main.cf):

```
# cd /usr/local/etc/postfix
# touch hello_access sender_access
# touch recipient_access client_access
# postmap hello_access
# postmap sender_access
# postmap recipient_access
# postmap client_access
```

Заполним недавно созданный каталог mysqlLookupMaps. В этом каталоге нам нужно создать следующие файлы:

- alias.conf — псевдонимы (листинг. 32.6);
- domain.conf — домены (листинг 32.7);
- mailbox.conf — почтовые ящики (листинг 32.8);
- quota.conf — квоты (листинг 32.9);
- sender.conf — отправители (листинг 32.10).

Листинг 32.6. Файл alias.conf

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
table = alias
select_field = goto
where_field = address
```

Листинг 32.7. Файл domain.conf

```
user = postfix
password = postfix
```

```
hosts = localhost
dbname = postfix
table = domain
select_field = domain
where_field = domain
additional_conditions = and active = '1' and backupmx = '0'
```

Листинг 32.8. Файл mailbox.conf

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
table = mailbox
select_field = maildir
where_field = username
additional_conditions = and active = '1
```

Листинг 32.9. Файл quota.conf

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
table = mailbox
select_field = quota
where_field = username
additional_conditions = and active = '1'
```

Листинг 32.10. Файл sender.conf

```
user = postfix
password = postfix
hosts = localhost
dbname = postfix
table = mailbox
select_field = username
where_field = username
additional_conditions = and active = '1'
```

Осталось только создать пользователя `virtual` и установить соответствующим образом права доступа:

```
# pw group add virtual -g 5555
# pw user add virtual -g virtual -s /sbin/nologin -u 5555
```

```
# mkdir /var/spool/mail/virtual
# chown virtual:virtual /var/spool/mail/virtual
# chmod 740 /var/spool/mail/virtual

# chown -R root:postfix /usr/local/etc/postfix/mysqlLookupMaps/
# chmod 440 /usr/local/etc/postfix/mysqlLookupMaps/*.conf
# chmod 550 /usr/local/etc/postfix/mysqlLookupMaps/
```

Обратите внимание на идентификатор пользователя и группы — `5555`, как и было указано ранее.

Все готово! Запускаем postfix:

```
# /usr/local/etc/rc.d/postfix start
postfix/postfix-script: starting the Postfix mail system
```

32.5. Установка PostfixAdmin

Осталось самое малое — установить интерфейс администратора PostfixAdmin, иначе управлять созданной нами конфигурацией будет не очень просто. Введите команды:

```
# cd /usr/ports/mail/postfixadmin
# make install clean
```

Откройте файл /usr/local/etc/apache22/httpd.conf и добавьте в него строки:

```
<Directory "/usr/local/www/postfixadmin/">
     Options none
     AllowOverride Limit
     Order Deny,Allow
     Deny from all
     Allow from 192.168.1.11
 </Directory>
```

Этим мы защитим PostfixAdmin — запустить его можно будет только с рабочей станции администратора (компьютер с адресом 192.168.1.11). Если есть желание запускать программу с других компьютеров сети, то можно задать IP-адрес сети, а не отдельного компьютера:

```
Allow from 192.168.1.
```

Создадим таблицы для PostfixAdmin:

```
# cd /usr/local/www/postfixadmin
# mysql -u root -p < DATABASE_MYSQL.TXT
```

Впрочем, перед выполнением сценария DATABASE_MYSQL.TXT (он содержит SQL-операторы, создающие все необходимые таблицы), я бы изменил пароли по умолчанию пользователей postfix и postfixadmin. Если вы измените пароли в файле DATABASE_MYSQL.TXT, не забудьте изменить их в следующих файлах:

- /usr/local/lib/sasl2/smtpd.conf;
- /usr/local/etc/authlib/authmysqlrc;

- /usr/local/etc/postfix/mysqlLookupMaps/alias.conf;
- /usr/local/etc/postfix/mysqlLookupMaps/domain.conf;
- /usr/local/etc/postfix/mysqlLookupMaps/mailbox.conf;
- /usr/local/etc/postfix/mysqlLookupMaps/quota.conf;
- /usr/local/etc/postfix/mysqlLookupMaps/sender.conf;
- /usr/local/www/postfixadmin/config.inc.php.

> ***Примечание***
>
> Если каким-то чудом файла DATABASE_MYSQL.TXT у вас не оказалось, вы всегда сможете его скачать с моего сайта: **http://dkws.org.ua/files/DATABASE_MYSQL.TXT**.

Пароли в файле DATABASE_MYSQL.TXT задаются с помощью функции `password()`, чтобы обеспечить шифрование пароля по алгоритму MD5:

```
# Postfix user & password
INSERT INTO user (Host, User, Password) VALUES
('localhost','postfix',password('postfix'));
…
# Postfix Admin user & password
INSERT INTO user (Host, User, Password) VALUES
('localhost','postfixadmin',password('postfixadmin'));
```

Теперь нужно отредактировать файл конфигурации PostfixAdmin /usr/local/www/postfixadmin/config.inc.php. В большинстве случаев вам понадобится изменить следующие параметры:

```
$CONF['database_type'] = 'mysql';          // тип БД
$CONF['database_host'] = 'localhost';              // имя сервера
$CONF['database_user'] = 'postfixadmin';   // имя пользователя
$CONF['database_password'] = 'postfixadmin';       // пароль
$CONF['database_name'] = 'postfix';        // база данных
```

Перезапустите Apache и обратитесь к каталогу postfixadmin:

http://mail.dkws.org/postfixadmin

В нашем случае (поскольку это первое обращение к PostfixAdmin) будет предложено перейти по ссылке **setup.php** (не забудьте удалить этот файл чуть позже).

Производим базовую настройку и переходим по ссылке **admin section** (она появится после настройки), вводим имя пользователя (`admin`) и пароль по умолчанию (пароль такой же — `admin`). Добавляем наш домен, нажав на кнопку **Добавить домен**. Затем добавляем почтовые ящики, нажав на кнопку **Добавить ящик**.

Далее управление сервером осуществляется с помощью кнопок и окошек — как в Windows. Полагаю, вы разберетесь с этим самостоятельно. Вместо объяснений, на какую кнопку нажимать, я вам подскажу, что делать дальше. А дальше можно "прикрутить" к нашей конфигурации Grey listing (мощнейший способ защиты от спама), о чем написано здесь: **http://www.postfix.org/SMTPD_POLICY_README.html**.

Описание настройки Grey Listing + postfix в этой книге не приводится, поскольку нельзя объять необъятное...



Глава 33

Удаленный доступ по протоколу SSH

33.1. Протокол SSH и SSH-клиент

Для сервера очень важна возможность удаленного доступа. Ведь не всегда есть возможность получить физический доступ к серверу — вы можете находиться в другом конце города или даже в другой стране, а сервер предприятия будет требовать вашего оперативного вмешательства.

Раньше для организации удаленного доступа к консоли сервера использовался протокол Telnet. В каждой сетевой операционной системе, будь то FreeBSD или Windows, имеется telnet-клиент. Эта программа так и называется — telnet (в Windows — telnet.exe).

После подключения с помощью telnet к удаленному компьютеру вы можете работать с ним, как будто он находится на вашем столе. В окне telnet-клиента вы увидите как бы консоль удаленного компьютера, вы будете вводить команды и получать результат их выполнения — все так, как если бы вы работали непосредственно за удаленным компьютером.

Но технологии не стоят на месте, и протокол Telnet устарел. Сейчас им практически никто не пользуется. Ему на смену пришел SSH (Secure Shell). SSH, как видно из названия, представляет собой безопасную оболочку. Главное отличие его от telnet состоит в том, что все данные (включая пароли доступа к удаленному компьютеру, а также пересылаемые по SSH файлы) передаются в зашифрованном виде. Во времена telnet участились случаи перехвата паролей и другой важной информации, что и стало причиной создания SSH.

SSH использует следующие алгоритмы для шифрования передаваемых данных: BlowFish, 3DES (Data Encryption Standard), IDEA (International Data Encryption Algorithm) и RSA (Rivest-Shamir-Adelman algorithm). Самыми надежными считаются алгоритмы IDEA и RSA. Поэтому, если вы передаете действительно конфиденциальные данные, лучше использовать один из этих алгоритмов.

В FreeBSD сервер OpenSSH (sshd) и клиент SSH (программа ssh) установлены по умолчанию, то есть для их установки вам не придется предпринимать никаких действий.

Если на вашей рабочей станции установлена Windows, и вам нужно подключиться к SSH-серверу, запущенному на BSD-машине, то по адресу **http://www.securitylab.ru/software/1478/** вы можете выбрать SSH-клиент для Widnows.

Работать с SSH-клиентом очень просто. Для подключения к удаленному компьютеру введите команду:

```
ssh [опции] <адрес_удаленного_компьютера>
```

В качестве адреса можно указать как IP-адрес, так и доменное имя компьютера. В табл. 33.1 приведены часто используемые опции программы ssh.

Таблица 33.1. *Опции программы ssh*

Опция	Описание
`-c blowfish|3des|des`	Используется для выбора алгоритма шифрования, при условии, что используется первая версия протокола SSH (об этом позже). Можно указать `blowfish`, `des` или `3des`
`-c шифр`	Задает список шифров, разделенных запятыми в порядке предпочтения. Опция используется для второй версии SSH. Можно указать `blowfish`, `twofish`, `arcfour`, `cast`, `des` и `3des`
`-f`	Переводит ssh в фоновый режим после аутентификации пользователя. Рекомендуется использовать для запуска программы X11. Например: `ssh -f server xterm`
`-l имя_пользователя`	Указывает имя пользователя, от имени которого нужно зарегистрироваться на удаленном компьютере. Опцию использовать не обязательно, поскольку удаленный компьютер и так запросит имя пользователя и пароль
`-p порт`	Определяет порт SSH-сервера (по умолчанию используется порт 22)
`-q`	"Тихий режим" — будут отображаться только сообщения о фатальных ошибках. Все прочие предупреждающие сообщения в стандартный выходной поток выводиться не будут
`-x`	Отключает перенаправление X11
`-X`	Задействовать перенаправление X11. Полезна при запуске X11-программ
`-1`	Использовать только первую версию протокола SSH
`-2`	Использовать только вторую версию протокола SSH. Вторая версия протокола более безопасна, поэтому при настройке SSH-сервера нужно использовать именно ее
`-4`	Заставляет SSH использовать только адреса IPv4
`-6`	Заставляет SSH использовать только адреса IPv6

33.2. SSH-сервер

Приступим к конфигурированию SSH-сервера. В большинстве случаев вы будете использовать сервер OpenSSH — открытую версию SSH-сервера (в FreeBSD другой и не будет, так как это свободная система).

Все настройки SSH-сервера хранятся в одном-единственном файле — /etc/ssh/sshd_config, а настройки программы-клиента — в файле /etc/ssh/ssh_config. Настройки программы клиента обычно задавать не нужно, поскольку они приемлемы по умолчанию. На всякий случай, вы можете заглянуть в файл /etc/ssh_config — его формат, как и назначение опций (большая часть из них закомментирована), вы поймете без моих комментариев.

Нас сейчас больше интересует файл sshd_config, содержащий конфигурацию SSH-сервера. Рассмотрим пример такого файла (листинг 33.1). Чтобы понять назначение директив, внимательно читайте комментарии, приведенные в листинге. Часть ненужных комментариев я удалил для сокращения объема листинга.

Листинг 33.1. Пример файла конфигурации /etc/ssh/sshd_config

```
# Задает порт, на котором будет работать SSH-сервер. Если директива
# не указана (закомментирована), то по умолчанию используется порт 22
#Port 22

# Директива Protocol позволяет выбрать версию протокола,
# рекомендуется использовать вторую версию
#Protocol 2,1
Protocol 2

# Директива AddressFamily задает семейство интерфейсов, которые должен
# прослушивать SSH-сервер
#AddressFamily any

# Локальный адрес, который должен прослушиваться SSH-сервером
#ListenAddress 0.0.0.0

# Ключевой файл для протокола SSH версии 1
HostKey /etc/ssh/ssh_host_key
# Ключевые файлы для второй версии протокола SSH
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key

# Время жизни ключа протокола первой версии. Время можно задавать в
# секундах или в часах (постфикс h, например, 1h — это 1 час или 3600
# секунд). По истечении указанного времени ключевой файл будет
# сгенерирован заново
#KeyRegenerationInterval 1h

# Разрядность ключа сервера в битах (только для первой версии протокола
# SSH)
```

```
#ServerKeyBits 1024

# Директивы управления протоколированием (можно не изменять)
#SyslogFacility AUTH
#LogLevel INFO

# Директивы аутентификации

# Время, предоставляемое клиенту для аутентификации. Задается в секундах
# или минутах (1m = 60 секунд). Если за это время клиент не
# аутентифицировал себя, соединение будет прекращено
#LoginGraceTime 2m

# Директива разрешает (yes) удаленный доступ пользователя root
PermitRootLogin yes

# Максимальное количество попыток аутентификации
#MaxAuthTries 6

# Использование RSA (yes)
#RSAAuthentication yes
# Аутентификация с открытым ключом (при значении yes)
#PubkeyAuthentication yes
#AuthorizedKeysFile   .ssh/authorized_keys

# Использование .rhosts-аутентификации с поддержкой RSA.
# Rhosts-аутентификацию использовать не рекомендуется, поэтому по
# умолчанию для этой директивы указано значение no. Если вы все-таки
# установите значение yes для этой директивы, то не
# забудьте добавить в файл /etc/ssh/ssh_known_hosts узлы,
# которым разрешен доступ к SSH-серверу. Только для первой
# версии протокола
#RhostsRSAAuthentication no

# Если вы используете вторую версию протокола и хотите разрешить
# Rhosts-аутентификацию, то вам нужно включить директиву
#HostbasedAuthentication,
# а разрешенные узлы указываются в файле ~/.ssh/known_hosts2
# HostbasedAuthentication no

# Если вы не доверяете пользовательским файлам ~/.ssh/known_hosts,
# установите значение yes для директивы IgnoreUserKnownHosts. Тогда будет
```

```
# использован только файл /etc/ssh/ssh_known_hosts
#IgnoreUserKnownHosts no

# Игнорировать файлы ~/.rhosts и ~/.shosts (рекомендуется установить yes)
#IgnoreRhosts yes

# Следующие директивы не рекомендуется изменять из соображений
# безопасности — они включают аутентификацию по паролю (а не IP-адресу
# компьютера, указанному в файле /etc/ssh/ssh_known_hosts)
# и запрещают использование пустых паролей
#PasswordAuthentication yes
#PermitEmptyPasswords no
```

После редактирования файла sshd_config добавьте следующую строку в файл /etc/rc.conf:

```
sshd_enable="YES"
```

Затем запустите SSH-сервер командой:

```
/etc/rc.d/sshd start
```

В конфигурационном файле наверняка вы заметили директивы указания файлов ключей. Ключи используются для аутентификации без паролей. Сгенерировать ключи можно утилитой `ssh-keygen`. Например, сгенерируем DSA-ключ:

```
# ssh-keygen -t dsa
```

Аналогично можно сгенерировать RSA-ключ:

```
# ssh-keygen -t rsa
```

Ключи RSA и DSA используются только во второй версии протокола SSH, для первой нужно генерировать ключ типа `rsa1`:

```
# ssh-keygen -t rsa1
```

Для большей безопасности файлы known_hosts и known_host2 необходимо хэшировать. Обеспечивает это опция `-H`:

```
# ssh-keygen -H
```

После хэширования файлов известных узлов вы уже не сможете отредактировать их вручную. Удалить хост из файла known_hosts (или known_host2), куда он попадает при создании ключа, можно опцией `-R`:

```
# ssh-keygen -R имя_узла
```

Исчерпывающее руководство по команде `ssh-keygen` вы найдете в руководстве:

```
man ssh-keygen
```

Для подключения к нашему серверу введите команду:

```
ssh 127.0.0.1
```

Можно также подключиться и с удаленного компьютера. Если сеть на локальном и удаленном компьютерах настроена правильно, проблем возникнуть не должно.

Глава 34



Web-сервер. Связка Apache + PHP + MySQL

34.1. Самый популярный Web-сервер

Apache — это Web-сервер с открытым исходным кодом. История его развития началась в 1995 году — тогда Apache был всего лишь "заплаткой", устраняющей ошибки популярного в то время Web-сервера NCSA HTTPd 1.3. Считается, что отсюда произошло и название Apache (a patchy — заплатка). Сейчас Apache — самый популярный Web-сервер в Интернете: в сентябре 2010 года было подсчитано, что он установлен на 56,6% Web-серверов[1].

Основные достоинства Apache — надежность, безопасность и гибкость настройки. Apache позволяет подключать различные модули, добавляющие в него новые возможности, — например, можно подключить модуль, обеспечивающий поддержку PHP или любого другого Web-ориентированного языка программирования.

Но есть и недостатки — без этого никак, всегда есть обратная сторона медали. Основной недостаток — отсутствие удобного графического интерфейса администратора. Да, настройка Apache осуществляется путем редактирования его конфигурационного файла. В Интернете можно найти простые конфигураторы Apache, но их возможностей явно не хватает для настройки всех функций Web-сервера.

34.2. Установка Web-сервера, интерпретатора PHP, сервера MySQL

В этой главе мы рассмотрим установку и настройку сервера Apache, интерпретатора PHP5 и сервера баз данных MySQL. Без всего этого сложно представить современный Web-сервер.

34.2.1. Установка Web-сервера Apache

Вы можете установить Apache как из портов, так и из пакета, содержащего уже откомпилированную версию сервера (напомню, что установка программного обеспечения была описана в *главе 18*). Я привык устанавливать программное обеспечение из портов. Да, это долго, но зато имеется возможность собрать программу

[1] См. **http://news.netcraft.com/archives/category/web-server-survey/**.

с нужными опциями, да и откомпилирована она будет на вашей машине, а, значит, с выполнением уж точно не будет никаких проблем. Ну, а желающие сэкономить время могут установить Apache из пакета. Далее все необходимое программное обеспечение будет устанавливаться из портов.

Об установке Apache в OpenBSD

В OpenBSD Apache поставляется сразу с системой — вам не придется его устанавливать. Только откройте файл /etc/rc.conf и установите значение "" для опции httpd_flags: `httpd_flags=""`. Этим вы обеспечите автоматический запуск Apache при старте системы.

Для установки Apache из портов введите команды:

```
# cd /usr/ports/www/apache22
# make install clean
```

Не забудьте перед вводом этих команд убедиться, что соединение с Интернетом доступно, иначе будет получено много недоуменных сообщений (рис. 34.1).

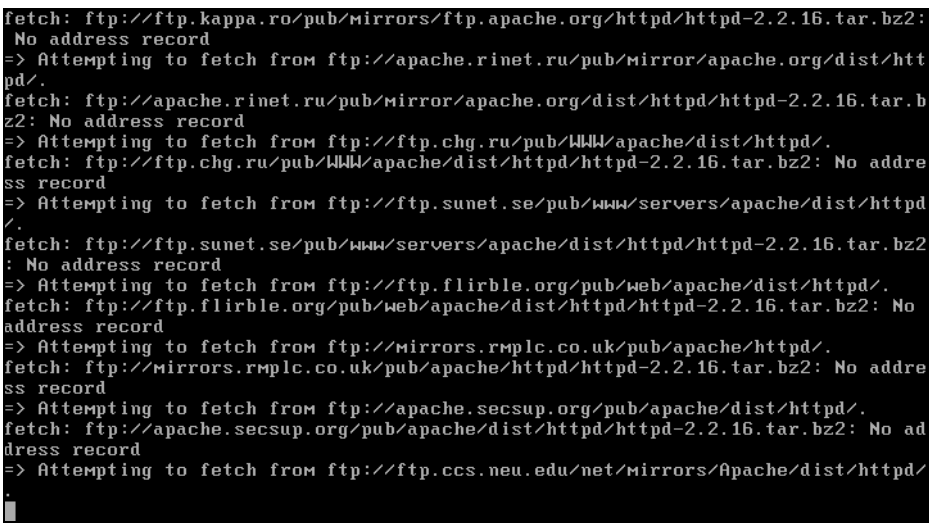


Рис. 34.1. Попытка установки из портов при неработающем интернет-соединении

Из-за чего ошибка?

Может быть, это и не относится непосредственно к установке Apache, но считаю нужным поделиться полезным опытом. Почему у меня возникла эта ошибка? Доступ к Интернету мой компьютер получает по локальной сети (сетевой интерфейс em0). При установке FreeBSD я выбрал экспресс-установку — в этом случае задается намного меньше вопросов, но пропускается настройка сети. Так вот, сеть-то я как раз и забыл настроить... Поэтому пришлось открыть файл /etc/rc.conf и добавить в него строку `ifconfig_em0="DHCP"` для организации DHCP-сервера (рис. 34.2). После этого нужно или перезагрузить компьютер (командой `reboot`), или перезагрузить сеть (командой `/etc/netstart`).

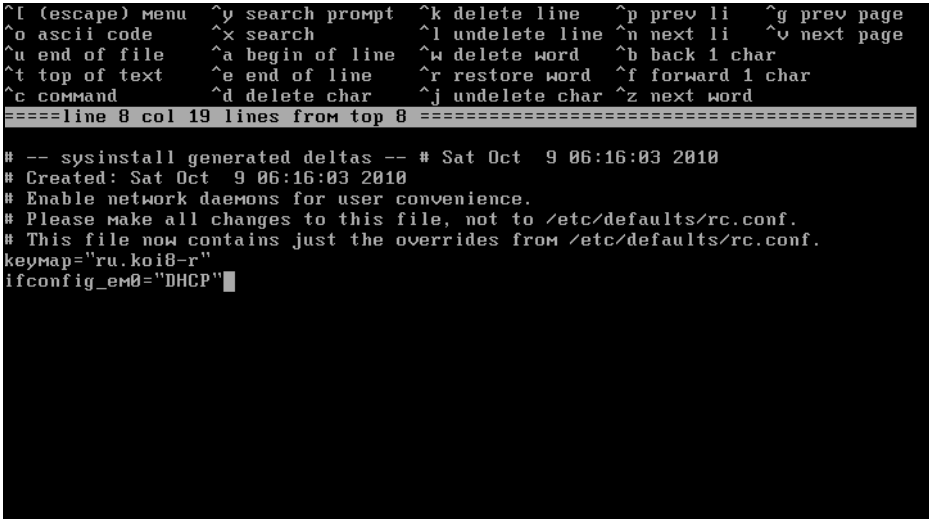


Рис. 34.2. Редактирование файла /etc/rc.conf

Совет

"Сервер и DHCP? — Это несерьезно!" — некоторые пользователи считают, что раз организуется сервер, то его сеть обязательно должна настраиваться вручную. Да, когда вы настраиваете первый сервер, который будет предоставлять всем остальным машинам доступ к сети, и на котором будет запущен DHCP-демон, тогда, конечно, приходится настраивать сеть вручную. А вот для всех остальных серверов это делать ни к чему! Просто пропишите в конфигурационном файле DHCP-сервера (с помощью блока `host`) IP-адреса для других серверов статически — с привязкой к MAC-адресу. Тогда при переустановке какго-либо сервера не нужно будет настраивать сеть.

Вернемся к нашим портам и будем считать, что с интернет-соединением у нас все в порядке. После ввода команды `make install clean` вы увидите окно, изображенное на рис. 34.3 (кстати, первая цель `install` компилирует и устанавливает программу, а вторая `clean` удаляет ненужные файлы, образовавшиеся в процессе компиляции программы). Здесь вы можете выбрать дополнительные опции. Если нет желания с ними разбираться, можете оставить все как есть и нажать кнопку **OK**, но я бы отключил за ненадобностью опцию IPV6 (поддержку протокола IP версии 6) — когда начнет использоваться IPv6, никто точно не знает, так зачем нам лишний код?

Далее начнется довольно долгий процесс сборки (рис. 34.4). Впрочем, от компьютера отходить не рекомендую, потому что система будет предлагать вам настроить различные дополнительные средства, например, Perl (рис. 34.5) и т. д. В большинстве случаев вам достаточно просто нажимать кнопку **OK**, но пока вы этого не сделаете, установка не будет продолжена (рис. 34.6).

Как вы уже успели понять, это далеко не все. Сам Web-сервер не представляет для нас никакой ценности. Нам еще нужны PHP 5 и MySQL.

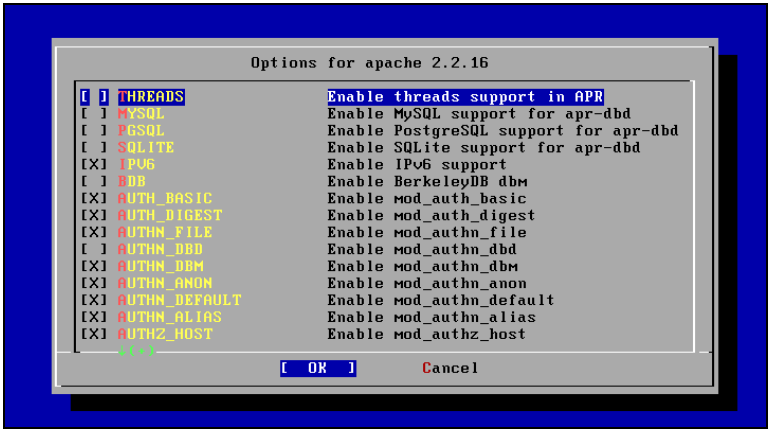


Рис. 34.3. Настройка порта apache 2.2.16

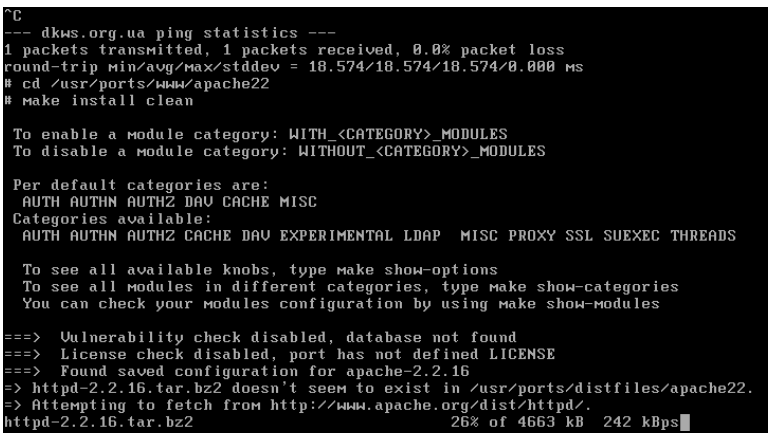


Рис. 34.4. Начата установка из портов

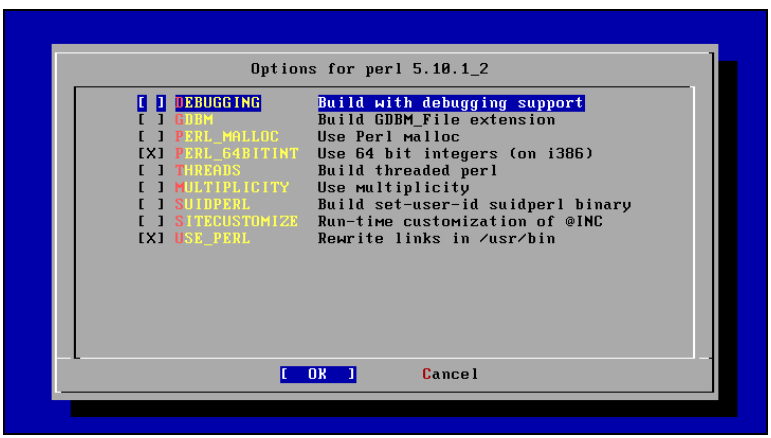


Рис. 34.5. Настройка Perl — просто нажмите кнопку OK

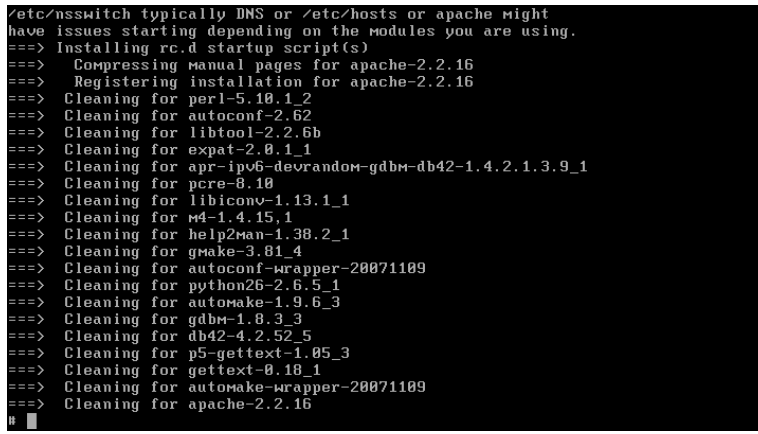


Рис. 34.6. Установка Apache завершена

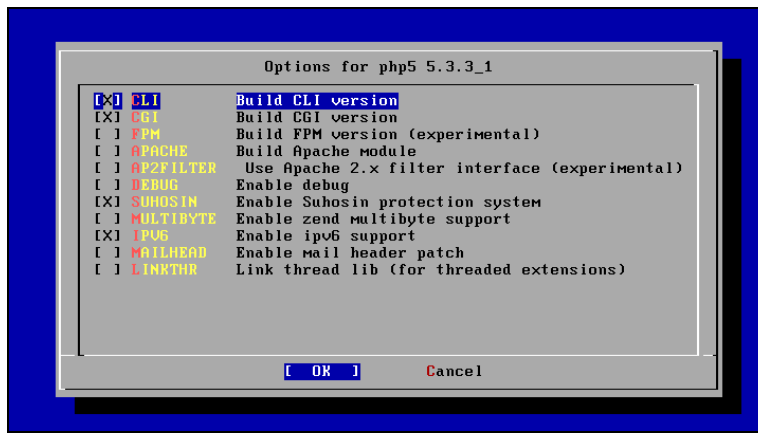


Рис. 34.7. Настройка порта PHP5

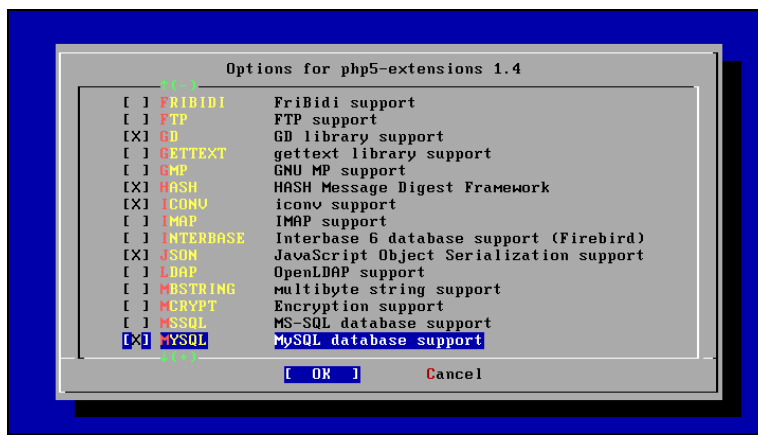


Рис. 34.8. Установка расширений PHP

34.2.2. Установка PHP

Для установки PHP введите команды:

```
# cd /usr/ports/lang/php5
# make install clean

# cd /usr/ports/lang/php5-extensions
# make install clean
```

> ***Внимание!***
>
> При установке порта `php5` обязательно выберите опцию `APACHE` — она добавляет поддержку PHP в Web-сервер Apache!

Первые две команды устанавливают сам PHP5. Вторые две — расширения PHP5. При установке PHP5 можно выключить IPv6 (рис. 34.7), если он вам точно не нужен (а если забудете, тоже ничего страшного).

А вот при установке расширений вам нужно обязательно выбрать `GD` и `MYSQL` (рис. 34.8) — эти расширения по умолчанию не устанавливаются, обязательно установите их! Первое расширение необходимо для поддержки интерпретатором PHP графических файлов — без него вы не сможете установить на свой сервер фотогалерею. А второе расширение — это поддержка MySQL. Остальные расширения можно выбрать по своему усмотрению.

Обрыв связи — реальная ситуация

Во время установки расширений PHP5 из портов мое интернет-соединение было оборвано. Как оказалось, провайдер менял оборудование. Соответственно, установка была прервана. Что делать в этом случае для возобновления установки? После восстановления соединения с Интернетом следует перейти в каталог с портом и ввести команды:

```
# make config
# make clean
# make install clean
```

Первая команда запустит окно настройки — вы сможете заново выбрать опции, если забыли, что выбирали ранее. Вторая команда удалит временные файлы, созданные во время первой попытки. Вторая команда установит порт и удалит временные файлы.

Установка PHP5 в OpenBSD

Подробно описывать установку PHP5 в OpenBSD смысла нет, укажу только основные команды:

```
# export PKG_PATH="ftp://ftp.openbsd.org/pub/OpenBSD/4.7/packages/i386/"
# pkg_add php5-core
# /usr/local/sbin/phpxs -s
```

Первая команда указывает путь, откуда загружать пакеты, вторая устанавливает PHP5, третья — добавляет модуль Apache.

34.2.3. Установка MySQL-сервера

После установки PHP самое время заняться установкой MySQL-сервера. Для его корректной установки нужно ввести четыре команды:

```
# cd /usr/ports/databases/mysql51-server
# make install clean
# mysqladmin -u root password новый_пароль
# /usr/local/bin/mysqld_safe &
```

> ***Примечание***
>
> В портах имеются версии MySQL 5.0 и 5.5. Для их установки нужно, соответственно, установить порты `/usr/ports/databases/mysql50-server` или `/usr/ports/databases/mysql55-server`.

Первые команды устанавливают сам MySQL-сервер. Затем происходит установка пароля для MySQL-пользователя root (не путайте MySQL-пользователя с системным пользователем root). Последняя команда запускает сервер MySQL прямо сейчас. Эту команду, впрочем, можно вообще не вводить. Ведь сейчас мы откроем файл /etc/rc.conf и добавим в него две строчки:

```
apache22_enable="YES"
mysql_enable="YES"
```

Как вы уже догадались, эти строки обеспечивают автоматический запуск Apache и MySQL при загрузке системы.

Далее можете или перезагрузить компьютер (командой `reboot`), или воспользоваться командами управления сервером *(см. разд. 34.3)* для запуска серверов Apache и MySQL. Но не спешите это делать. Сначала нужно подготовить Apache к первому запуску. Для этого откройте основной конфигурационный файл Apache — /usr/local/etc/apache22/httpd.conf:

```
# ee /usr/local/etc/apache22/httpd.conf
```

Найдите в нем директиву `ServerName`. Ее значение — имя вашего Web-сервера. Укажите реальное имя, зарегистрированное в DNS. Если у вас еще не настроен DNS-сервер, пропишите имя сервера в файле /etc/hosts, иначе сервер при запуске будет "ругаться".

34.3. Управление серверами Apache и MySQL

Чуть ранее мы обеспечили автоматический запуск серверов Apache и MySQL. Но иногда нужен перезапуск в процессе работы, например, после изменения конфигурационных файлов. Для управления Apache можно использовать утилиту `apachectl`:

```
# /usr/local/sbin/apachectl start
# /usr/local/sbin/apachectl restart
# /usr/local/sbin/apachectl stop
```

Первая команда запускает сервер (если он остановлен), вторая — перезапускает (полезно после изменения конфигурации сервера), третья — останавливает сервер Apache.

Для управления Apache можно также использовать скрипт /usr/local/etc/rc.d/apache22:

```
# /usr/local/etc/rc.d/apache22 start
# /usr/local/etc/rc.d/apache22 start
# /usr/local/etc/rc.d/apache22 stop
```

Однако лично мне больше нравится команда `apachectl` — ее можно вводить без длинного пути, а вот скрипт apache22 — нельзя.

Для управления MySQL-сервером используются команды:

```
# /usr/local/etc/rc.d/mysql-server start
# /usr/local/etc/rc.d/mysql-server restart
# /usr/local/etc/rc.d/mysql-server stop
```

34.4. Проблемы с запуском Apache

Здесь мы рассмотрим ряд проблем, возникающих при запуске Apache. Могу уверить вас, что в других книгах по FreeBSD вы не найдете решения подобных проблем.

Итак, после долгожданной установки Apache, PHP5 и расширений PHP5 я ввел команду запуска Apache. Результат ее выполнения показан на рис. 34.9. Я получил следующие сообщения:

httpd: apr_sockaddr_info_get() failed for

httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1 for ServerName

…Failed to enable the 'httpready' Accept Filter

Первые две ошибки связаны с тем, что я забыл установить директиву `ServerName` (хотя "разбор полетов" с ней вам еще обеспечен!).

```
===>   php5-extensions-1.4 depends on file: /usr/local/lib/php/20090626/xmlwrite
r.so - found
===>   Generating temporary packing list
===>  Checking if lang/php5-extensions already installed
===>   Registering installation for php5-extensions-1.4
===>  Cleaning for php5-posix-5.3.3_1
===>  Cleaning for php5-session-5.3.3_1
===>  Cleaning for php5-simplexml-5.3.3_1
===>  Cleaning for php5-tokenizer-5.3.3_1
===>  Cleaning for php5-xml-5.3.3_1
===>  Cleaning for php5-xmlreader-5.3.3_1
===>  Cleaning for php5-xmlwriter-5.3.3_1
===>  Cleaning for php5-extensions-1.4
# apachectl start
httpd: apr_sockaddr_info_get() failed for
httpd: Could not reliably determine the server's fully qualified domain name, us
ing 127.0.0.1 for ServerName
[Sat Oct 09 17:00:49 2010] [warn] (2)No such file or directory: Failed to enable
 the 'httpready' Accept Filter
# apachectl stop
httpd: apr_sockaddr_info_get() failed for
httpd: Could not reliably determine the server's fully qualified domain name, us
ing 127.0.0.1 for ServerName
httpd (no pid file) not running
#
```

Рис. 34.9. Ошибки при запуске Apache

Третья ошибка более серьезная. Лечится она командой:

```
kldload accf_http
```

Чтобы не вводить эту команду каждый раз, добавьте следующую строку в файл /boot/loader.conf:

```
accf_http_load="YES"
```

Хорошо, установил директиву `ServerName`, отредактировав файл /etc/hosts (поскольку не хотел изменять настройки DNS-сервера), и перезагрузил компьютер командой `reboot`.

При загрузке увидел сообщение о невозможности запуска Apache. Смотрю в журнал ошибок /var/log/httpd-error.log. Вижу знакомое сообщение (рис. 34.10):

hostname nor servname provided, or not known: mod_unique_id: unable to find IPv4 address of "" Configuration Failed

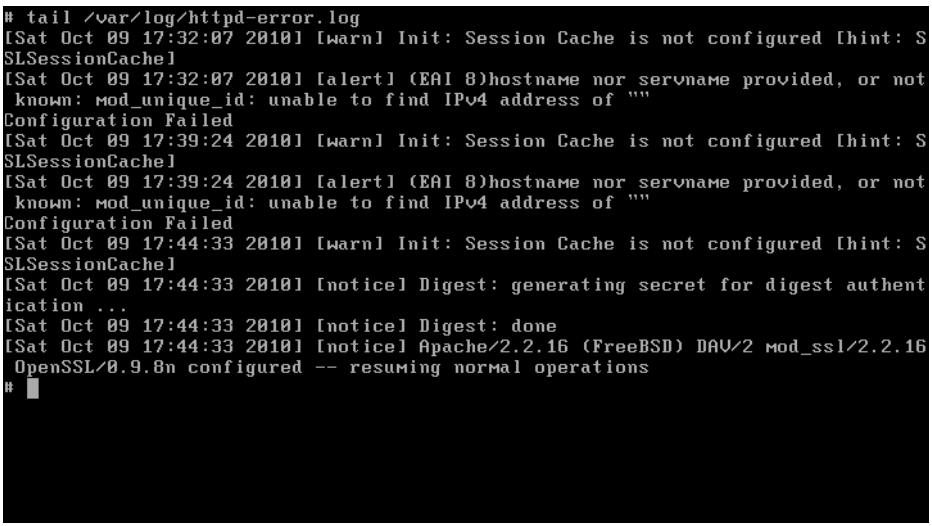


Рис. 34.10. Журнал ошибок Apache

Пробую в качестве значения `ServerName` указать IP-адрес — не помогло, потом — адрес 127.0.0.1, потом вообще закомментировал `ServerName` (адрес 127.0.0.1 используется по умолчанию) — ничего не помогло. Тогда я решил избавиться от модуля `mod_unique_id` — открыл файл httpd.conf и закомментировал в нем строку:

```
LoadModule unique_id_module libexec/apache22/mod_unique_id.so
```

После этого сервер запустился без проблем, о чем свидетельствует последнее сообщение в журнале ошибок (см. рис. 34.10).

34.5. Тестирование настроек

Осталось самое малое — протестировать нашу связку Apache + PHP + MySQL. Убедитесь, что серверы Apache и MySQL запущены, — если вы внесли изменения в файл /etc/rc.conf, для этого достаточно перезагрузить компьютер.

Создайте сценарий test.php (листинг 34.1) и поместите его в каталог /usr/local/www/apache22/data (или другой каталог, заданный директивой `DocumentRoot`).

Листинг 34.1. Файл test.php

```
<?
phpinfo();
?>
```

Затем введите в строке браузера адрес `http://имя/test.php`, здесь `имя` — это то, что вы указали в директиве `ServerName`. Тестировать можно как с другой машины сети, так и с локальной машины, но тогда вам придется установить текстовые браузеры links и/или lynx.

Для установки текстового браузера links введите команды:

```
# cd /usr/ports/www/links
# make install clean
```

Чтобы установить lynx, используются вот эти команды:

```
# cd /usr/ports/www/lynx
# make install clean
```

Вы должны увидеть тестовую страницу PHP (рис. 34.11). Если вы вместо этого получили:

- сообщение о недоступности сервера — убедитесь, что сервер запущен (команда `apachectl`) и что вы правильно указали имя сервера;
- код PHP-файла вместо результата работы — неправильно установлена связка Apache + PHP, повторите все действия после установки Apache. Наиболее вероятная причина — вы забыли отметить опцию `APACHE` при сборке порта php5 (рис. 34.12).

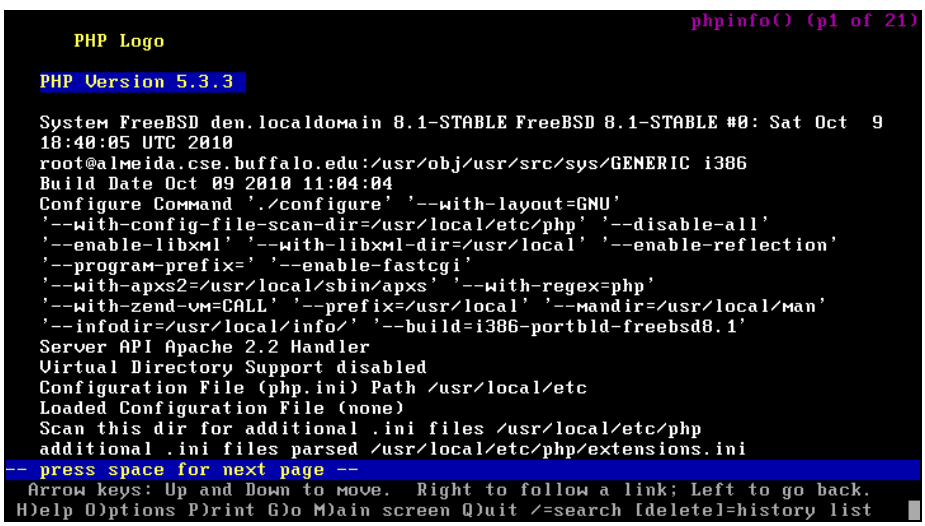


Рис. 34.11. Тестовая страница PHP

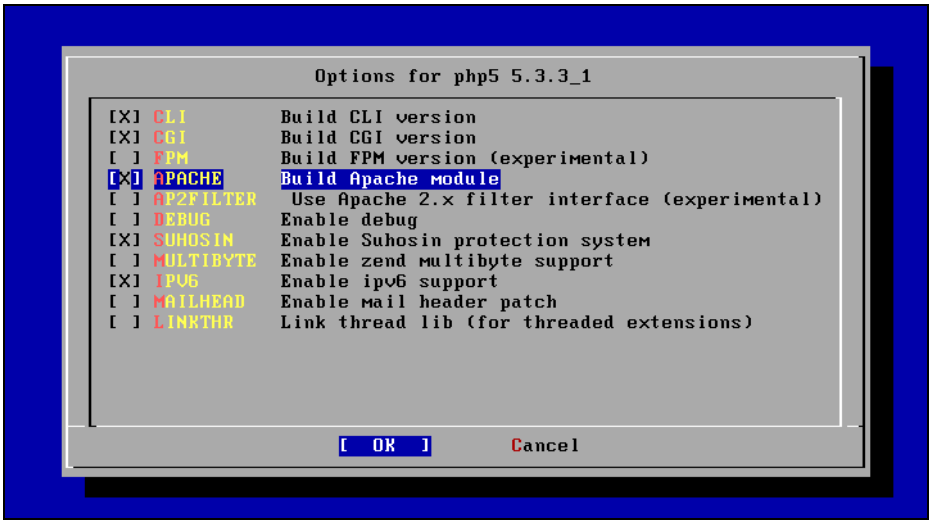


Рис. 34.12. Сборка порта php5

34.6. Файлы конфигурации Web-сервера

34.6.1. Базовая настройка

Конфигурационные файлы Apache находятся в каталоге /usr/local/etc/apache22. В табл. 34.1 приведены названия конфигурационных файлов Apache версии 2.2.

Таблица 34.1. *Конфигурационные файлы Apache 2.2*

Файл/каталог	Описание
httpd.conf	Основной файл конфигурации
mime.types	Содержит описание различных MIME-типов
magic	Содержит подсказки для модуля mod_mime_magic, который помогает серверу Apache распознать MIME-тип файла
каталог Includes	Содержит дополнительные конфигурационные файлы с необязательными параметрами, вынесенными в отдельные файлы, чтобы не засорять основной конфигурационный файл
каталог extra	Содержит файлы с экстра-настройками. Эти настройки выходят за рамки базовой конфигурации сервера, поэтому вынесены в отдельные файлы. Например, в файле httpd-vhosts.conf находятся настройки виртуальных узлов, а в файле httpd-mpm.conf — параметры, влияющие на производительность сервера

С основным конфигурационным файлом мы уже знакомы (редактировали его) и знакомы с директивой `ServerName`. В *разд. 34.6.2* мы познакомимся с другими директивами.

34.6.2. Самые полезные директивы файла конфигурации

Понятно, что для полноценной настройки сервера одной директивы `ServerName` недостаточно. В табл. 34.2 приведены самые полезные директивы файла конфигурации Apache. Нужно отметить, что в таблице не рассматриваются некоторые директивы (например, `Port`, `BindAddress`), которые не используются во второй версии Apache.

Таблица 34.2. *Директивы файла конфигурации*

Директива	Описание
`ServerName имя`	Задает имя Web-сервера, имя должно быть зарегистрированным на DNS-сервере, то есть обычно это доменное имя сервера
`ServerAdmin e-mail`	Задает e-mail администратора сервера
`ServerRoot каталог`	Определяет каталог с конфигурационными файлами сервера
`PidFile файл`	Определяет имя файла, в котором будет храниться PID исходного процесса Web-сервера. Обычно изменять эту директиву не нужно

`Listen`	Задает адрес интерфейса (и порт, если нужно), на котором будет работать Apache. Три примера использования `Listen`:
	`Listen 192.168.1.1`
	`Listen 192.168.1.1:80`
	`Listen 80`
	В первом случае мы указали адрес интерфейса 192.168.1.1. Запросы, поступаемые с других интерфейсов, Apache обрабатывать не будет. Во втором случае мы еще и уточнили порт сервера, хотя делать этого не нужно было, так как порт 80 используется по умолчанию. В третьем случае мы просто указали порт (используется для задания номера порта, отличного от 80). Не вижу особого смысла в использовании `Listen`. Если вам нужно, чтобы сервер был доступен только в локальной сети, но не был доступен интернет-пользователям, правильнее бы его закрыть брандмауэром, а не указывать адрес локального интерфейса в `Listen`

`DocumentRoot каталог`	Позволяет задать каталог, в котором хранятся документы Web-сервера — это корневой каталог документов. Обычно это /usr/local/www/apache22/data
`StartServers N`, `MaxSpareServers N`, `MinSpareServers N`, `MaxClients N`	Директивы, непосредственно влияющие на производительность сервера. Мы их рассмотрим отдельно в *разд. 34.7*

Таблица 34.2 *(продолжение)*

Директива	Описание
`KeepAlive On | Off`, `KeepAliveTimeout N`	Управляют постоянными соединениями, будут рассмотрены в *разд. 34.7*
`DirectoryIndex` список	Задает имена файлов, которые могут использоваться в качестве главной страницы (индекса). Значение по умолчанию `index.html index.cgi index.pl index.php index.xhtml` Учитывая это значение, сервер сначала будет искать в каталоге файл index.html, потом index.cgi и т. д.
`HostnameLookups On|Off`	Если директива включена (`On`), то IP-адрес клиента перед записью в журнал будет разрешен (то есть Web-сервер вычислит доменное имя клиента перед записью информации о попытке доступа в журнал). Выключение (`Off`) этой опции позволяет повысить производительность сервера, поскольку не нужно тратить время на разрешение IP-адресов в доменные имена
`ErrorLog` файл	Задает журнал ошибок. Значение по умолчанию /var/log/httpd-error.log. Обычно не нужно изменять это значение
`TransferLog` файл	Задает журнал обращений к серверу. Значение по умолчанию /var/log/httpd-access.log. Обычно не нужно изменять это значение
`Timeout N`	Тайм-аут в секундах (время, на протяжении которого сервер будет ждать возобновления прерванной попытки передачи данных)
`User` пользователь `Group` группа	Директивы `User` и `Group` задают имя пользователя и группы, от имени которых запускается Web-сервер
`FancyIndexing on | off`	Если пользователь в запросе не укажет имя документа, а только каталог, но в нем не окажется главной страницы, заданной директивой `DirectoryIndex`, сервер передаст пользователю оглавление каталога. Данная директива определяет, в каком виде будет передано оглавление каталога: в более красивом, со значками каталогов и описаниями файлов (значение `On`), или в более простом (`Off`). Позже мы заберемся, как запретить вывод содержимого каталога (из соображений безопасности)

`Redirect`	Позволяет создавать перенаправление с одного каталога на другой или даже на другой сервер. Например:
	`Redirect /orders /billing`
	`Redirect /den http://dkws.org.ua`
	В первом случае при обращению к каталогу /orders будет выполнено перенаправление на каталог /billing. А во втором случае при обращении к каталогу /den будет выполнен редирект на сайт http://dkws.org.ua

`AddIcon` картинка список	Если `FancyIndexing` включена, то `AddIcon` позволяет связать графическую картинку с типом файла, например, `AddIcon /images/graphics.gif .gif, .jpeg, .bmp, .png, .tiff`

***Таблица 34.2** (окончание)*

Директива	Описание
`DefaultIcon картинка`	Позволяет задать картинку по умолчанию (`AddIcon`, `FancyIndexing`)
`ErrorDocument N файл`	Позволяет задать файл, содержащий сообщение об ошибке, для ошибки с номером *N*, например: `ErrorDocument 404 /errors/file_not_found.html`
`Directory, Limit, Location, Files`	Это так называемые *блочные* директивы, которые нельзя описать одной строкой, поэтому о них мы поговорим отдельно (*см. разд. 34.6.3*)

34.6.3. Директивы *Directory*, *Limit*, *Location*, *Files*

Рассмотрим сначала блочные директивы `Directory` и `Limit`.

- С помощью блочной директивы `Directory` можно установить параметры отдельного каталога. Внутри директивы `Directory` могут использоваться директивы `AllowOverride`, `Limit`, `Options`. Вот пример определения параметров корневого сервера:

```
<Directory />
AllowOverride None
Options None
</Directory>
```

Значения `None` для обеих директив (`AllowOverride` и `Options`) считаются самыми безопасными. `None` для `AllowOverride` запрещает использование файлов .htaccess, которые могут переопределять директивы конфигурационного файла Apache. К тому же, `AllowOverride None` позволяет повысить производительность сервера.

Допустимые опции каталога (значения директивы `Options`) указаны в табл. 34.3.

***Таблица 34.3.** Опции каталога*

Опция	Описание
`None`	Запрещены все опции
`All`	Все опции разрешены
`Indexes`	Если указана эта опция, при отсутствии файла, заданного `DirectoryIndex`, будет выведено оглавление каталога. Если `Options` установлена в `None` (или `Indexes` не указана в списке опций), то оглавление каталога выводиться не будет. Можно запретить только вывод оглавления каталога, тогда вместо значения `None` нужно установить значение `Indexes`
`Includes`	Разрешает использование SSI (Server Side Includes)

Таблица 34.3 *(окончание)*

Опция	Описание
`IncludesNoExec`	Более безопасный режим SSI: разрешает SSI, но запрещает запускать из включений внешние программы
`ExecCGI`	Разрешает выполнение CGI-сценариев
`FollowSymLink`	Разрешает использование символических ссылок. Довольно опасная опция, поэтому лучше ее не использовать
`SymLinksIfOwnerMatch`	Похожа на предыдущую, но разрешает следовать ссылке, если объект, на который она указывает, принадлежит тому же пользователю, что и ссылка

- Блочная директива `Limit` позволяет ограничить доступ. Внутри этой директивы можно использовать директивы `order`, `deny` и `allow` (вообще есть еще и директива `require`, но она очень редко используется). Директива `order` задает порядок выполнения директив `deny` и `allow`:

```
# сначала запретить, потом разрешить
order deny, allow
# сначала разрешить, потом запретить
order allow, deny
```

Директивы `allow` и `deny` нужно использовать так:

```
# запрещаем доступ всем
deny from all
# разрешаем доступ только нашей сети
allow from firma.ru
```

Пример использования директив `Directory` и `Limit` представлен в листинге 34.2.

Листинг 34.2. Фрагмент файла конфигурации Apache

```
<Directory />
AllowOverride None
Options None
<Limit>
       order deny, allow
       # запрещаем доступ всем
       deny from all
       # разрешаем доступ только нашей сети
       allow from firma.ru
</Limit>

</Directory>
```

Вместо доменного имени firma.ru можно указать адрес сети, например, "192.168.1." (без кавычек, кавычки здесь нужны, чтобы показать наличие точки после единицы).

В качестве параметра директиве `Limit` можно передать метод передачи данных (`GET`, `POST`), например:

```
<Limit GET>
<Limit POST>
```

Теперь обратимся к блочным директивам `Location` и `Files`.

- Директива `Location` очень похожа на директиву `Directory`. Только если `Directory` ограничивает доступ к каталогу, то `Location` предназначена для ограничения доступа к отдельным URL сервера:

```
<Location URL>
директивы ограничения доступа
</Location>
```

К директивам ограничения доступа относятся `order`, `deny`, `allow`.

- Директива `Files` предназначена для ограничения доступа к отдельным файлам:

```
<Files файл>
директивы ограничения доступа
</Files>
```

Вы можете указать как отдельный файл, так и регулярное выражение, которому должны соответствовать файлы:

```
# запрещаем доступ к файлу privat.html всем, кроме нашей сети
<Files privat.html>
order deny, allow
deny from all
allow from firma.ru
</Files>
# запрещаем доступ к файлам .ht* всем
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
</Files>
```

Файлы .htaccess заслуживают отдельного внимания. Директивы этих файлов могут переопределять параметры сервера, однако после изменения файла .htaccess не нужно перезагружать сервер. Представим, что вы предоставляете хостинг нескольким десяткам пользователей. Каждый пользователь хочет определить свои уникальные параметры, например, каждый хочет ограничить доступ к своему каталогу с помощью блока `Directory`. Разрешив использованием .htaccess, вы спасаете себя от постоянных обращений пользователей с просьбами изменить httpd.conf. Поверьте, это очень напрягает. Но не забывайте правильно установить параметры использования файлов .htaccess директивой `AllowOverride`. Она может принимать следующие значения:

- `None` — полное игнорирование файлов .htaccess. Идеально подходит, если вы используете сервер в гордом одиночестве. Тогда отказ от .htaccess повысит производительность сервера;

- `All` — разрешает переопределять в .htaccess все параметры сервера. Забудьте об этом значении из соображений безопасности;
- `Options` — в .htaccess можно использовать только директиву `Options`;
- `Limit` — в .htaccess можно использовать только директиву `Limit`;
- `FileInfo` — в .htaccess можно использовать директивы `AddType` и `AddEncoding`;
- `AuthConfig` — в .htaccess можно использовать директивы аутентификации (`AuthName`, `AuthType` и др.).

Формат файла .htaccess чем-то напоминает формат httpd.conf. Можно сказать, что это мини-файл конфигурации. Вот пример использования `Limit` в .htaccess:

```
<limit GET, POST>
order deny,allow
deny from all
allow from localhost
</limit>
```

Доступ к каталогу, в котором находится этот конфигурационный файл, возможен только с узла localhost. Это идеально походит для каталога, в котором вы собираетесь отлаживать ваши PHP-сценарии.

Кстати, для отладки PHP-сценариев добавьте в файл .htaccess следующие строки:

```
php_flag display_errors on
php_value error_reporting 7
php_flag register_globals on
```

Первая включит отображение ошибок, вторая установит максимальный уровень отчета об ошибках, а третья включит директиву интерпретатора PHP `register_globals`. Помните, что эти параметры небезопасны, а походят только для отладки сценариев!

Мы рассмотрели все самые полезные директивы конфигурационного файла Apache. Напомню, что директивы, непосредственно влияющие на производительность сервера, будут рассмотрены в *разд. 34.7*.

34.7. Оптимизация Apache

В конфигурационных файлах extra/httpd-defaults.conf и extra/httpd-mpm.conf вы найдете параметры, влияющие на производительность сервера. Правильно установив их, вы можете повысить производительность Apache.

Так, в файле extra/httpd-mpm.conf имеется директива `MaxClients`, позволяющая ограничить число одновременно работающих клиентов. Чтобы правильно установить это значение, нужно знать, сколько пользователей может одновременно зайти на сервер. При небольшой посещаемости вполне хватит значения 30–50, при большой загрузке количество одновременно работающих клиентов может исчисляться сотнями. Следите за посещаемостью вашего сервера и корректируйте это значение, иначе какая-то часть пользователей может остаться "за бортом", а им это очень не понравится (или же, наоборот, ресурсы сервера будут использоваться нерационально).

Директива `StartServers` задает количество экземпляров сервера, которые будут созданы при запуске исходной копии сервера. Для этой директивы можно устано-

вить значение, равное 10% от `MaxClients`. Устанавливать большое значение не нужно, поскольку вы будете нерационально использовать ресурсы компьютера.

Рассмотрим обычную ситуацию. Для `MaxClients` вы установили значение 200, а для `StartServers` — 20. Запросы первых 20 клиентов будут обрабатываться очень быстро, поскольку сервисы уже запущены. Запрос 21 клиента будет обслужен чуть медленнее, поскольку придется запустить еще одну копию Apache. И, тем не менее, не нужно устанавливать в нашем случае (`MaxClients = 200`) для `StartServers` значение больше 20 — ведь не всегда даже 20 человек одновременно заходят на сервер. Если же на сервере постоянно находится как минимум 20 человек, тогда нужно увеличить значения и `MaxClients`, и `StartServers`.

Впрочем, бывают и исключения, например, для сервера внутренней корпоративной сети. В этом случае вы точно знаете, сколько клиентов имеется в вашей сети, следовательно, можно точно знать, какое значение установить для `MaxClients` и `StartServers`. Но, все равно, для `MaxClients` нужно установить чуть большее значение, чем для `StartServers` — на всякий случай:

```
MaxClients 150
StartServers 100
```

Чтобы еще эффективнее оптимизировать работу Web-сервера, нужно понимать, как он работает: клиент посылает запрос, Web-сервер его обрабатывает и посылает клиенту ответ. После этого соединение можно закрывать и завершать копию Apache, обслуживающую это соединение. Но зачем завершать копию Web-сервера, если сейчас же на сайт зайдет другой пользователь, и опять нужно будет запускать еще одну копию сервера, что только увеличивает загрузку процессора. Поэтому с помощью директивы `MaxSpareServers` можно установить максимальное число серверов, которые останутся в памяти уже после закрытия соединения с пользователем, — они будут просто ждать своего пользователя. Теоретически, чтобы сбалансировать нагрузку, значение для `MaxSpareServers` можно установить такое же, как и для `StartServers`, то есть 10% от `MaxClients`.

Вы не задумывались, что если Web-сервер будет работать в режиме постоянного соединения, то это повысит его производительность? Если вы об этом подумали, то мыслите в правильном направлении. Представим, что у нас на сайте есть форум. Человек редко заходит на форум, чтобы посмотреть одну страничку. Обычно он может находиться на форуме часами. Так зачем же закрывать соединение? Чтобы потом опять тратить время и ресурсы на его открытие? Разрешить постоянные соединения можно с помощью директивы `KeepAlive`. Она задает максимальное число таких соединений:

```
KeepAlive 5
```

А директива `KeepAliveTimeout` задает тайм-аут для постоянного соединения в секундах:

```
KeepAliveTimeout 15
```

Директива `KeepAliveRequests` позволяет ограничить число запросов за одно постоянное соединение:

```
MaxKeepAliveRequests 100
```

Все директивы настройки постоянных соединений находятся в файле extra/httpd-defaults.conf.

Используя все упомянутые в этом разделе директивы, вы сможете добиться существенного повышения производительности вашего Web-сервера.

34.8. Пользовательские каталоги

Если вы когда-нибудь настраивали сервер Apache, то наверняка знакомы с директивой `UserDir`. Я специально ее не описал в табл. 34.2, потому что она заслуживает отдельного разговора.

По умолчанию директива `UserDir` отключена:

```
UserDir disabled
```

Включить ее можно, указав вместо `disabled` любое другое значение — обычно указывается значение `public_html`:

```
UserDir public_html
```

Затем в пользовательском каталоге /home/<имя> создается каталог public_html, в него помещаются HTML/PHP-файлы персонального сайта пользователя. Обращение к сайту пользователя происходит по URL:

http://имя_сервера/~имя_пользователя

Например, если при включенной директиве `UserDir` вы поместили в каталог /home/den/public_html файл report.xml, то обратиться к нему можно по адресу:

http://server/~den/report.xml

Директива `UserDir disabled` позволяет указать имена пользователей, которым запрещено иметь собственные каталоги Web-сервера:

```
UserDir disabled root toor daemon operator
```

Параметры, относящиеся к пользовательским каталогам, вы найдете в файле конфигурации extra/httpd-userdir.conf. В этом каталоге можно будет задать директиву `UserDir`, а также определить параметры доступа к пользовательским каталогам (рис. 34.13).

Поддержка пользовательских каталогов возможна только при включенном модуле `mod_userdir`. Взглянем на фрагмент файла httpd.conf (рис. 34.14). Можно видеть, что модуль `mod_userdir` включен (директива `LoadModule`), но файл конфигурации extra/httpd-userdir.conf по умолчанию не подключается. Чтобы включить возможность использования пользовательских каталогов, вам нужно открыть файл httpd.conf и раскомментировать строку:

```
#Include etc/apache22/extra/httpd-userdir.conf
```

После этого нужно перезапустить сам Apache:

```
# apachectl restart
```

Поздравляю! Вы только что создали простейший хостинг. Осталось только, как описано в *главе 30*, настроить FTP-сервер, чтобы пользователи могли удаленно обращаться к своим каталогам. Если вам такого сервера мало и есть еобходимость в использовании разных доменных имен для каждого пользовательского каталога,

тогда вам нужно настроить виртуальные серверы. Директивы, относящиеся к виртуальным узлам, находятся в файле /extra/httpd-vhosts.conf[1].

На этом настройку нашего Web-сервера можно считать завершенной. В *главе 36* мы поговорим о защите Apache, а пока можете приступить к созданию виртуальной частной сети *(см. главу 35)*.

```
#
# Required module: mod_userdir

#
# UserDir: The name of the directory that is appended onto a user's home
# directory if a ~user request is received.  Note that you must also set
# the default access control for these directories, as in the example below.
#
UserDir public_html
UserDir disabled root toor daemon operator bin tty kmem games news man sshd bind
 proxy _pflogd _dhcp uucp pop www nobody mailnull smmsp
#
# Control access to UserDir directories.  The following is an example
# for a site where these directories are restricted to read-only.
#
<Directory "/home/*/public_html">
    AllowOverride FileInfo AuthConfig Limit Indexes
    Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
    <Limit GET POST OPTIONS>
        Order allow,deny
        Allow from all
    </Limit>
    <LimitExcept GET POST OPTIONS>
        Order deny,allow
:
```

Рис. 34.13. Вывод клманды `cat extra/httpd-userdir.conf | less`

```
# cat httpd.conf ¦ grep userdir
LoadModule userdir_module libexec/apache22/mod_userdir.so
#Include etc/apache22/extra/httpd-userdir.conf
#
```

Рис. 34.14. Фрагмент httpd.conf, относящийся к UserDir

[1] А сам процесс настройки описан в моей статье "Виртуальность": **http://www.dkws.org.ua/index.php?page=show&file=a/servers/vs**.



Глава 35

Виртуальные частные сети

35.1. Для чего нужна виртуальная частная сеть?

Предположим, что пользователям нашей организации нужно обращаться к ресурсам корпоративной сети, когда они находятся за ее пределами, например, в другом городе. Первое, что приходит в голову, — это настроить сервер удаленного доступа (Remote Access Server, RAS или dial-in сервер). Пользователь с помощью модема дозванивается к серверу удаленного доступа, сервер идентифицирует пользователя, после чего последний подключается к сети предприятия и работает в ней — как обычно (разве что скорость передачи данных будет значительно ниже, чем обычно).

Но использование RAS — затея довольно дорогая и неудобная. Во-первых, придется организовать модемный пул, а это недешево и накладно: понадобятся или многоканальная линия, или же несколько телефонных линий (чтобы обеспечить одновременную работу нескольких пользователей). Во-вторых, нужно оплачивать междугородние и даже международные звонки пользователей (для удобства самих пользователей надо будет организовать callback-режим). В-третьих, далеко не всегда у пользователя есть возможность подключиться к телефонной сети. В-четвертых, RAS не может обеспечить связь нескольких филиалов компании.

Выходом из сложившейся ситуации является использование виртуальной частной сети (Virtual Private Network, VPN). В случае с VPN данные передаются по каналам Интернета. Это существенно упрощает и удешевляет нашу задачу. Доступ к Интернету есть везде, пользователи сами смогут выбирать провайдера и способ (соответственно, и скорость) подключения к Интернету. Понятно, чтобы оградить себя от перехвата информации, данные при передаче через VPN шифруются. Вот основные преимущества VPN:

- не потребуется никакого дополнительного оборудования (модемного пула) и каких-либо дополнительных ресурсов (например, многоканальной телефонной линии). Понадобится только подключение к Интернету, а поскольку нет такого частного предприятия, которое не было бы подключено к Интернету, будем считать, что все необходимое для организации VPN уже есть;
- безопасность передачи данных по сравнению с обычной передачей данных по Интернету;

- возможность как соединения филиалов компании, так и подключения отдельных пользователей к корпоративной сети. При этом мобильные пользователи могут заходить в Интернет по каналам 3G/EDGE, что делает подключение к VPN максимально гибким — им не придется искать свободную телефонную розетку.

35.2. Необходимое программное обеспечение

Настроить VPN можно разными средствами:

- для организации соединений типа сеть-сеть, то есть для связи двух сетей одной компании в единую VPN, мы будем использовать IPsec. Впрочем, можно использовать также и OpenVPN;
- для подключения к корпоративной сети удаленных пользователей мы будем использовать протокол PPTP (Point to Point Tunneling Protocol). Однако эту же задачу можно решить средствами IPsec и OpenVPN.

IPsec считается более безопасным, но PPTP намного проще настраивается, в чем вы успеете убедиться в этой главе.

35.3. Соединение сеть-сеть

35.3.1. Постановка задачи

Предположим, что нам нужно связать два офиса компании. Один будет находиться в Москве, другой — во Владивостоке (Киеве, Санкт-Петербурге, да хоть в другом районе Москвы — разницы нет).

В каждом офисе есть компьютер под управлением FreeBSD, который является шлюзом по умолчанию для своих сетей. Текущая конфигурация обеих сетей (до настройки VPN) изображена на рис. 35.1: компьютер с адресом 94.94.94.94 является шлюзом для всей сети 192.168.1.0 (как вы догадались, 192.168.1.1 — это внутренний IP-адрес шлюза), а компьютер с адресом 95.95.95.95 является шлюзом для сети 192.168.2.0 (192.168.2.1 — это внутренний IP-адрес второго шлюза). Наша задача — объединить сети 192.168.1.0 и 192.168.2.0.

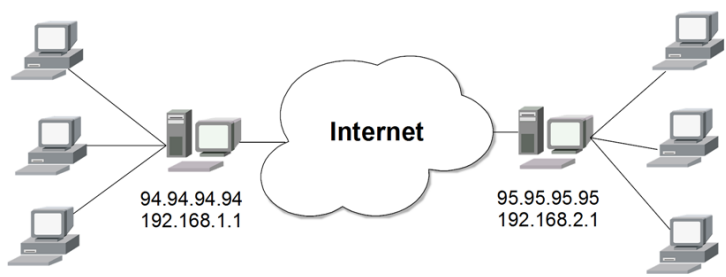


Рис. 35.1. Конфигурация сети

35.3.2. Выбор канала передачи данных

Для связи офисов будут использоваться VPN-маршрутизаторы. В роли такого маршрутизатора может выступать любой UNIX-компьютер с установленным программным обеспечением (Ipsec, в данном случае).

Важно правильно выбрать канал для подключения VPN-маршрутизатора к Интернету. Канал не должен быть "узким", иначе данные между филиалами компании будут передаваться очень медленно. Обычные выделенные линии, понятно, отпадают. Также отпадают различные беспроводные соединения, типа RadioEthernet. Конечно, беспроводное соединение — это удобно, но его качество очень сильно зависит от зашумленности эфира и от погоды — если на улице плохая погода, скорость заметно падает, не говоря уже о том, что беспроводная точка доступа может сгореть во время грозы (да, чуть ли не после каждой грозы вам придется менять точку доступа). А выключить точку доступа на время непогоды может себе позволить отдельный пользователь, но не предприятие.

Действительную надежность и независимость могут беспечить синхронные (двунаправленные) спутниковые соединения, но оборудование, да и содержание такого канала (лицензия на передатчик, оплата) будут совсем не дешевым удовольствием. Если организация может себе его позволить, уверен, она не пожалеет о таком решении.

Оптимальным вариантом для многих организаций будет использование DSL-соединений. Они обеспечивают вполне приличную скорость обмена данными — до нескольких мегабит в секунду, да и надежность их несравнима с беспроводными (имеется в виду RadioEthernet, а не спутник).

Если вы остановили свой выбор на DSL-соединении, рекомендую выбирать вариант SDSL (синхронное) — скорость приема и передачи данных будет одинаковой. Существуют также и ADSL-соединения (асинхронные), но скорость приема у них в несколько раз ниже скорости передачи. Такие соединения больше подходят для домашнего использования, чем для связи офисов компании.

35.3.3. Перекомпиляция ядра

Думаю, вы уже привыкли, что при настройке чего-то серьезного в FreeBSD приходится перекомпилировать ядро. На этот раз нам нужны следующие опции:

```
options IPFIREWALL
options IPFIREWALL_DEFAULT_TO_ACCEPT
options IPFIREWALL_VERBOSE
options IPFIREWALL_VERBOSE_LIMIT=1000
options TCP_DROP_SYNFIN
options IPSEC
options IPSEC_ESP
```

Примечание

Если вы перекомпилировали ядро при настойке шлюза и последовали моим рекомендациям по включению опций `IPSEC` и `IPSEC_ESP` *(см. разд. 28.2)*, то на данный момент ваше ядро должно быть полностью готовым "к бою".

Понятно, что ядро нужно перекомпилировать на обоих серверах. Практически все действия следует производить параллельно — с одной стороны и с другой. Если есть помощник — хорошо, если нет — настраивайте доступ по SSH *(см. главу 33)* и работайте удаленно.

Чтобы не терять времени, можете производить всю настройку параллельно перекомпиляции ядра. Только в процессе настройки не перезагружайте машину и ничего не запускайте. Перезагрузитесь уже после сборки ядра.

35.3.4. Установка ipsec-tools

После перекомпиляции ядра нужно установить порт ipsec-tools — именно порт, поскольку при его сборке мы можем указать нужные нам опции `DEBUG`, `DPD`, `FRAG` (рис. 35.2):

```
# cd /usr/ports/security/ipsec-tools/
# make install clean
```

Примечание

Да, после установки в FreeBSD графической среды GNOME *(см. главу 9)* выходить из нее не хочется, поэтому не могу не продемонстрировать на рис. 35.2 свой графический терминал.

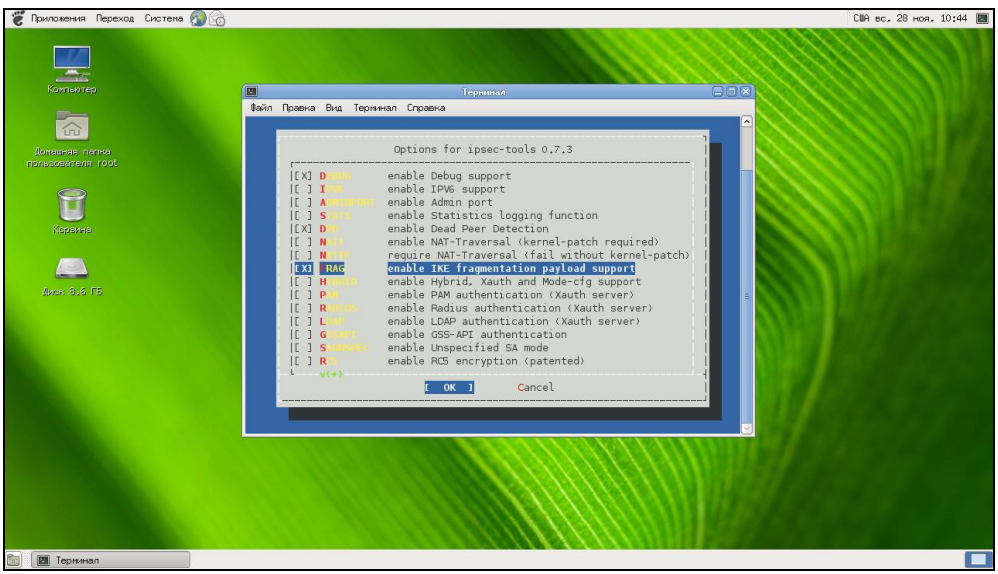


Рис. 35.2. Сборка ipsec-tools

35.3.5. Генерирование сертификатов

После сборки ipsec-tools наступает, наверное, самый ответственный момент — генерирование сертификатов. Сертификаты будут находиться в каталоге /usr/local/etc/racoon/cert. Если у вас нет такого каталога, создайте его:

```
# mkdir -p /usr/local/etc/racoon/cert
```

Заодно скопируем пример конфигурационного файла:

```
# cd /usr/local/etc/racoon
# cp /usr/local/share/examples/ipsec-tools/racoon.conf racoon.conf
```

Сертификаты нужно создать для обоих серверов. Пусть первый сервер называется server.left.com, а второй — gate.right.com.

> ***Совет***
>
> Вообще-то имена файлов не важны — можете вместо них взять IP-адреса или придумать условные названия. Но если вы станете использовать доменные имена, вам будет проще ориентироваться.

Сгенерируем сертификат для сервера server.left.com (это длинная команда, и она должна поместиться в одну строку):

```
# openssl req -new -nodes -newkey rsa:1024
 -sha1 -keyform PEM -keyout server.left.com.private -outform PEM
 -out server.left.com.pem
```

Перед тем как вводить команды создания ключа, рекомендую ознакомиться с их выводом. Вам нужно будет ответить на ряд несложных вопросов (ответы выделены шрифтом `Courier`):

Generating a 1024 bit RSA private key
.....++++++
..++++++
writing new private key to 'server.left.com.private'

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]:`RU`
State or Province Name (full name) [Some-State]:`Russia`
Locality Name (eg, city) []:`Moscow`
Organization Name (eg, company) [Internet Widgits Pty Ltd]:`Home Ltd`
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:`Denis`
Email Address []:`root@server.left.com`

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:

В результате будут созданы файлы server.left.com.pem и server.left.com.private. Теперь сгенерируем еще файл server.left.com.public (этот файл генерируется на базе двух только что созданных, поэтому их нужно указать в командной строке):

```
# openssl x509 -req -in server.left.com.pem -signkey server.left.com.private
-out server.left.com.public
```

Теперь надо проделать то же самое на второй машине, только укажите другие имена файлов (для узла gate.right.com).

После того как сертификаты будут созданы на обеих машинах, обменяйтесь public-сертификатами. То есть скопируйте файл server.left.com.public на машину gate.right.com (в каталог /usr/local/etc/racoon/cert) и наоборот — файл gate.right.com.public на машину server.left.com.

35.3.6. Редактирование файлов конфигурации

Откройте файл /etc/rc.conf первой машины и добавьте в него строки:

```
# Запускаем racoon
racoon_enable="YES"

# Создаем интерфейс для тоннеля
cloned_interfaces="gif0"
gif_interfaces="gif0"
gifconfig_gif0="94.94.94.94 95.95.95.95"
ifconfig_gif0="inet 192.168.1.1 192.168.2.1 netmask 255.255.255.0"

# Запускаем ipsec
ipsec_enable="YES"

# вводим статический роутинг
static_routes="AnotherLan"
route_AnotherLan="192.168.2.0/24 -interface gif0"
```

Приведенная конфигурация актуальна для первой машины (192.168.1.1). Для второй машины (192.168.2.0) следует поменять местами все IP-адреса:

```
gifconfig_gif0="95.95.95.95 94.94.94.94"
ifconfig_gif0="inet 192.168.2.1 192.168.1.1 netmask 255.255.255.0"
route_AnotherLan="192.168.1.0/24 -interface gif0"
```

Затем откройте на первой машине файл /etc/ipsec.conf и добавьте в него строки:

```
spdadd 94.94.94.94/32 95.95.95.95/32 ipencap -P out ipsec
 esp/tunnel/94.94.94.94-95.95.95.95/require;
spdadd 95.95.95.95/32 94.94.94.94/32 ipencap -P in ipsec
 esp/tunnel/95.95.95.95-94.94.94.94/require;
```

Содержимое этого файла на второй машине будет таким же, только нужно поменять местами IP-адреса.

Осталось отредактировать файл /usr/local/etc/racoon/racoon.conf (листинг 35.1).

Листинг 35.1. Файл /usr/local/etc/racoon/racoon.conf

```
# Путь к файлам конфигурации
path include "/usr/local/etc/racoon" ;
# Путь к сертификатам
path certificate "/usr/local/etc/racoon/cert/" ;

# Уровень протоколирования:
# "notify", "debug" или "debug2"
log debug2;

# Блок "padding" задает параметры формирования пакетов
# Изменять его (без знания, что вы делаете) не рекомендуется!
padding
{
        maximum_length 20;
        randomize off;
        strict_check off;
        exclusive_tail off;
}

# Определяем, какие интерфейсы слушать. Для второй машины установите ее
# адрес!
listen
{
        #isakmp ::1 [7000];
        isakmp 94.94.94.94 [500];
}

# Таймеры (можно не изменять)
timer
{
        counter 5;
        interval 20 sec;
        persend 1;
        phase1 30 sec;
        phase2 15 sec;
}

# Описываем второй компьютер (95.95.95.95), на второй машине все
# аналогично, только нужно изменить IP-адреса и имена сертификатов
remote  95.95.95.95
```

```
{
        exchange_mode aggressive,main;
        my_identifier asn1dn;
        peers_identifier asn1dn;
       # сертификаты первого сервера, все должно быть в одну строку!
certificate_type x509 "server.left.com.public" "server.left.com.private";
       # сертификат удаленного сервера
        peers_certfile x509 "gate.right.com.public";
        proposal {
                encryption_algorithm 3des;
                hash_algorithm sha1;
                authentication_method rsasig;
                dh_group 2 ;
        }
}

# Необходимая секция, без нее получите ошибку при запуске racoon
sainfo anonymous
{
        pfs_group 5;
        lifetime time 60 min;
        encryption_algorithm 3des ;
        authentication_algorithm hmac_sha1;
        compression_algorithm deflate ;
}
```

Перезагружаемся и вводим команду `ifconfig`. Кроме прочих интерфейсов в выводе `ifconfig` должен присутствовать интерфейс gif0:

gif0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 1280
tunnel inet 94.95.94.94 --> 95.95.95.95
inet 192.168.1.1 --> 192.168.2.1 netmask 0xffffffff

Практически все, задача выполнена, тоннель между сетями создан. Но чтобы все работало, как мы того ожидаем, нужно настроить брандмауэр. Если брандмауэры настроены в режиме `OPEN` *(см. главу 28)*, то все заработает сразу, и вы сможете пропинговать удаленный сервер по его внутреннему адресу — 192.168.2.1:

```
# ping 192.168.2.1
```

PING 192.168.2.1 (192.168.2.1): 56 data bytes
64 bytes from 192.168.2.1: icmp_seq=0 ttl=64 time=2.160 ms
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=2.255 ms

Кстати, до перезагрузки лучше настроить брандмауэр в режиме `OPEN`, чтобы после перезагрузки убедиться, что тоннель работает. После этого следует переконфи-

гурировать брандмауэр в обычный режим и добавить правила (на второй машине укажите IP-адрес первой машины):

```
${fwcmd} add allow ip from any to any via gif0
${fwcmd} add allow udp from 95.95.95.95 to me dst-port 500
${fwcmd} add allow esp from me to 95.95.95.95
${fwcmd} add allow esp from 95.95.95.95 to me
```

35.4. Соединение клиент-сеть

Здесь мы рассмотрим настройку соединения клиент-сеть, когда нужно обеспечить подключение удаленного отдельного пользователя к внутренней локальной сети предприятия. Для настройки соединения такого типа используется протокол PPTP. Удаленный клиент сможет через Интернет попасть в корпоративную сеть (рис. 35.3) и использовать все ее преимущества: сетевые принтеры, корпоративные серверы и т. п.

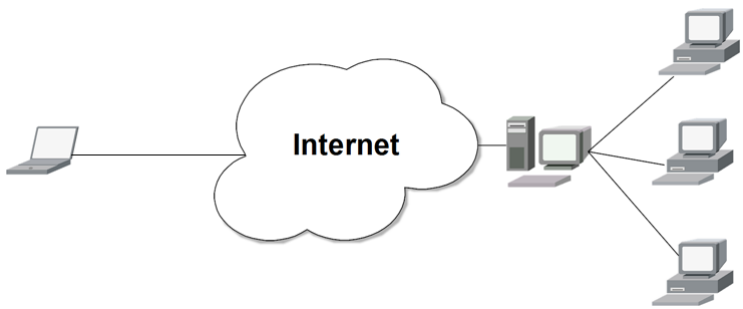


Рис. 35.3. Соединение клиент-сеть

35.4.1. Выбор канала передачи данных

Удаленный пользователь может сам выбрать тот вид соединения, которое для него предпочтительно. В большинстве случаев мобильные пользователи в командировках будут использовать каналы 3G/EDGE. Да, недостатков у 3G/EDGE достаточно (хотя намного меньше, чем у GPRS), но зато подключиться к родной сети можно будет практически откуда угодно — мобильный телефон всегда под рукой, лишь бы находиться в зоне покрытия мобильного оператора.

35.4.2. Перекомпиляция ядра

Проверьте, есть ли в конфигурационном файле ядра сервера корпоративной сети следующие устройства:

```
pseudo-device  ppp     1       # Kernel PPP
pseudo-device  tun             # Packet tunnel
```

Если они закомментированы, раскомментируйте их и перекомпилируйте ядро. Пока ядро собирается, можете продолжить настройку, только ничего не запускайте и не перезагружайте компьютер до завершения компиляции ядра.

35.4.3. Установка порта poptop

Установим порт poptop:

```
# cd /usr/ports/net/poptop
# make install clean
```

35.4.4. Редактирование конфигурационных файлов

Откройте файл /usr/local/etc/pptpd.conf. Удалите все, что в нем есть, и добавьте следующие строки:

```
options /etc/ppp/options.pptpd
debug
noipparam
```

Когда все будет настроено и протестировано, параметр `debug` можно будет удалить — он нужен только для отладки.

Файл /etc/ppp/options.pptpd нужно довести до такого состояния:

```
proxyarp
+MSChap-V2 mppe-128 mppe-stateless
```

Затем отредактируйте файл /etc/ppp/ppp.conf (листинг 35.2).

Листинг 35.2. Файл /etc/ppp/ppp.conf

```
pptp:
 enable proxy
 set dns 95.111.111.7        # IP-адрес DNS-сервера
 set ifaddr 192.168.1.1      # внутренний адрес сервера
 set timeout 300             # тайм-аут простоя: к-во секунд до разрыва
                             # соединения, 0 — соединение не разрывается
                             # вообще

 enable MSChapV2             # протокол шифрования
```

Теперь создадим файл /etc/ppp/ppp.secret, в который нужно поместить имена и пароли PPTP-пользователей (удаленных клиентов) и IP-адреса внутренней сети, которые будут им присвоены, когда те войдут в сеть:

```
denis  123456 192.168.1.101
user   123456 192.168.1.102
```

Формат этого файла прост: имя пользователя, пароль, IP-адрес.

В файл /etc/rc.conf добавьте строку:

```
pptpd_enable="YES"
```

Осталось только дождаться перекомпиляции ядра. Если ядро уже перекомпилировано (допустим, вы это сделали заранее), тогда запустите `pptpd`:

```
/usr/local/etc/rc.d/pptpd.sh start
```

В заключение настроим брандмауэр, добавив следующие правила:

```
allow tcp from any to me 1723
allow gre from any to any
allow ip from any to any via tun0
```

Указанные правила нужно дописать после правил NAT, иначе VPN работать не будет.

На этом все. Настройка PPTP-сервера завершена. Осталось только настроить клиенты.

35.5. Настройка PPTP-клиентов

35.5.1. Настройка Linux-клиента

Здесь мы рассмотрим настройку клиента, работающего под управлением Linux.

1. Для установки VPN-клиента установите пакеты pptp-linux или pptp-client. После установки запустите сценарий `pptp-command`. Сценарий отобразит меню из четырех опций (рис. 35.4) — выберите опцию **Setup**.
2. Вы увидите меню, из которого следует выбрать опцию **Manage CHAP secrets** (см. рис. 35.4), а затем — **Add a New CHAP secret**. Сценарий попросит ввести имя локальной машины, имя удаленной машины (вводить необязательно), имя пользователя и пароль.

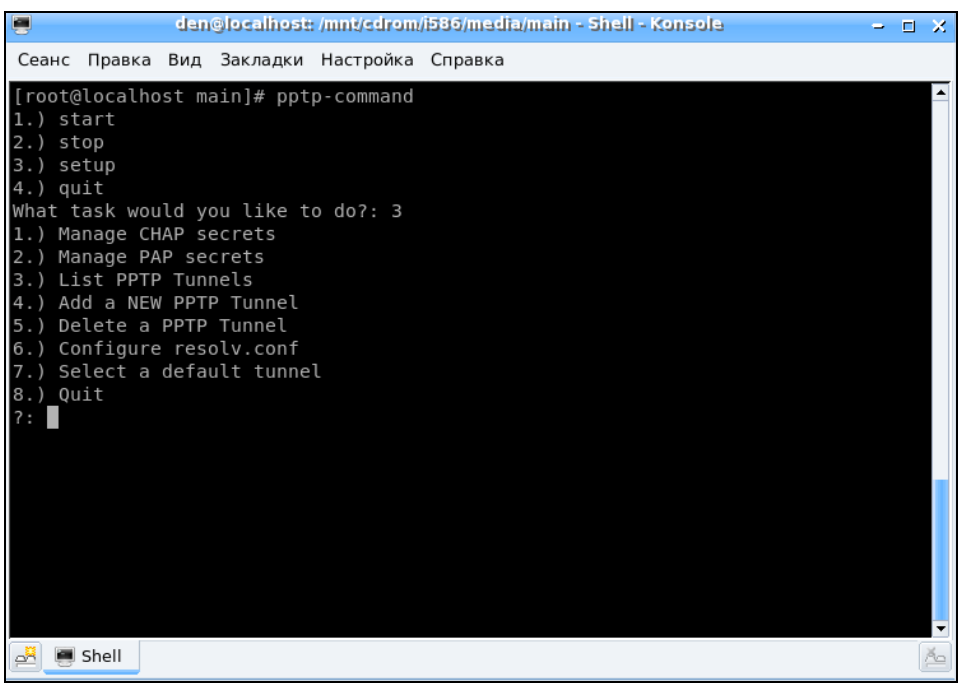


Рис. 35.4. Сценарий `pptp-command`

3. Вы вернетесь в меню настройки. Выберите теперь опцию **Add a new PPTP tunnel**, а затем — **Other**. Вам будет предложено ввести следующую информацию: имя и IP-адрес VPN-сервера, а также параметры маршрутизации.
4. Выберите опцию **Configure resolv.conf** и укажите IP-адреса DNS-серверов.
5. Настройка почти закончена. В уже хорошо знакомом вам меню выберите опцию **Select a default tunnel**, позволяющую выбрать туннель по умолчанию. Выберите туннель, который только что создали.
6. Для подключения к VPN нужно опять запустить сценарий `pptp-command` и выбрать опцию **Start**. Понятно, что перед этим вы должны подключиться к Интернету.

В Ubuntu Linux в окне **Сетевые соединения** (NetworkManager) имеется вкладка **VPN**, но кнопка **Добавить** неактивна. А все потому, что не установлены пакеты, реализующие поддержку VPN. Чтобы настроить VPN-соединение через NetworkManager, вам нужно скачать с сайта **packages.ubuntu.com** пакеты network-manager-pptp и network-manager-pptp-gnome. Конечно, если вы используете не протокол PPTP, а какой-нибудь другой, тогда придется скачать и установить соответствующие пакеты, например, network-manager-vpnc.

Чтобы не запутать вас окончательно, приведу список пакетов, которые нужно скачать и установить (включая пакеты, от которых зависят необходимые пакеты):

- pptp-linux;
- network-manager-pptp;
- network-manager-pptp-gnome;
- network-manager-vpnc;
- network-manager-openvpn;
- network-manager-openvpn-gnome;
- network-manager-strongswan.

Первые три пакета нужны для поддержки PPTP, четвертый — для протокола VPNC (Cisco), следующие два — для протокола OpenVPN, последний — для strongSwan.

А как же скачать пакеты, если соединение с Интернетом осуществляется по VPN? Тут три варианта:

- перезагрузиться в Windows, если она установлена, подключиться к Интернету, скачать пакеты;
- найти другой компьютер или использовать альтернативное соединение (например, 3G-модем или мобильный телефон);
- использовать дистрибутив Denix (**http://denix.dkws.org.ua**) — в него по умолчанию включены все пакеты, необходимые для установки VPN-соединений.

Ради справедливости нужно отметить, что в Ubuntu 10.04[1] появилась поддержка VPN "из коробки", но поддерживается только PPTP, а вот если нужно установить соединение по другому протоколу — см. действия, описанные ранее.

[1] Если у вас все-таки Ubuntu 9.*x* и переходить на 10.*x* вы пока не планируете, обязательно прочитайте мою статью по настройке VPN-соединения в Ubuntu 9: **http://www.dkws.org.ua/index.php?page=show&file=a/ubuntu/vpn-ubuntu9**.

Скачанные пакеты можно установить командой `dpkg` (подробно об этом вы прочитаете в соответствующей главе любой книги по Linux).

Процесс настройки VPN-соединения в Linux Mandriva подробно описан по адресу: **http://wiki.mandriva.com/ru/Настройка_VPN_в_Mandriva**[1].

35.5.2. Настройка Windows-клиента

Настройка VPN-подключения в Windows 2000/XP намного проще. Во-первых, вам не нужно устанавливать VPN-клиент — он уже входит в состав Windows и установлен по умолчанию. Во-вторых, интерфейс мастера новых подключений в Windows дружелюбнее и привычнее интерфейса `pptp-command` (хотя кому как).

Чтобы создать VPN-подключение, выполните команду меню **Пуск | Настройка | Сетевые подключения | Создание нового подключения**. В окне мастера новых подключений выберите **Подключить к сети на рабочем месте** (рис. 35.5), а затем — **Подключение к виртуальной частной сети** (рис. 35.6).

Перед подключением к VPN нужно установить соединение с Интернетом, поэтому мастер новых подключений предложит вам выбрать соединение с Интернетом, которое будет установлено перед подключением к VPN (рис. 35.7).

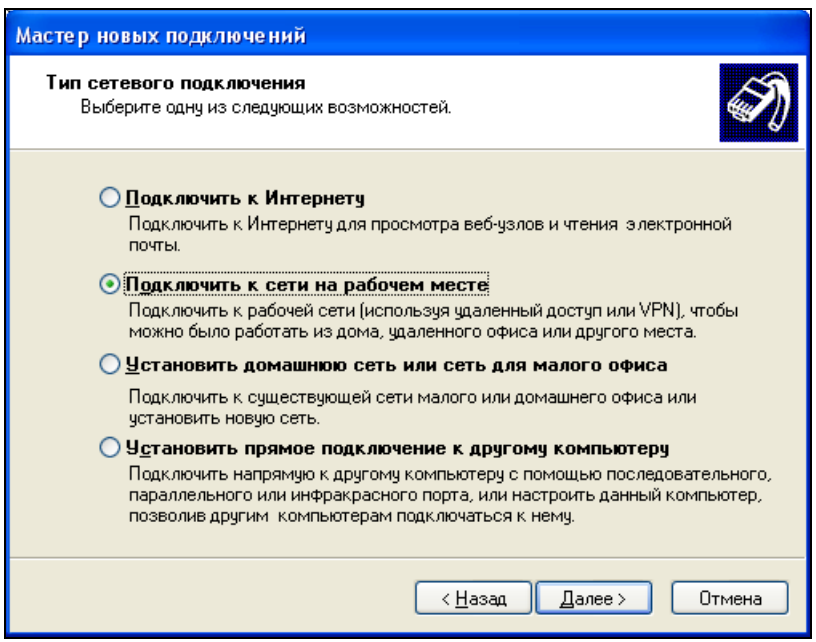


Рис. 35.5. Мастер новых подключений

[1] Тем не менее, при настройке VPN-соединения в Mandriva могут возникнуть некоторые проблемы. Решение ряда проблем описано здесь: **http://www.dkws.org.ua/phpbb2/viewtopic.php?p=18432**.

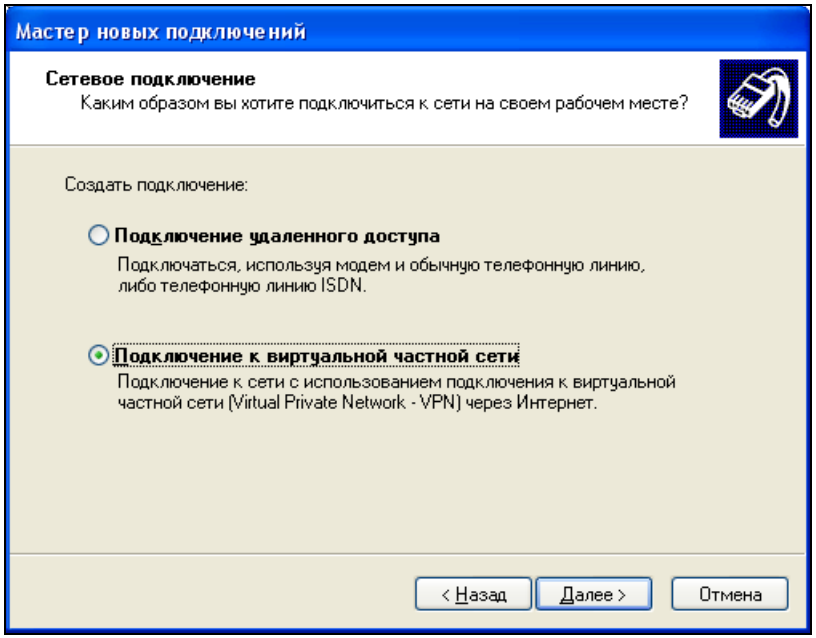


Рис. 35.6. Подключение к VPN

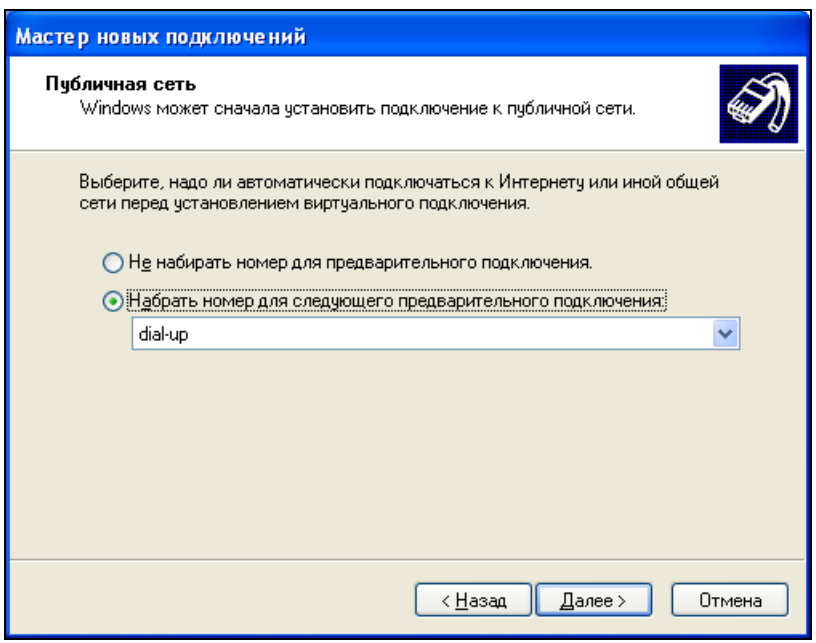


Рис. 35.7. Выбор подключения к Интернету

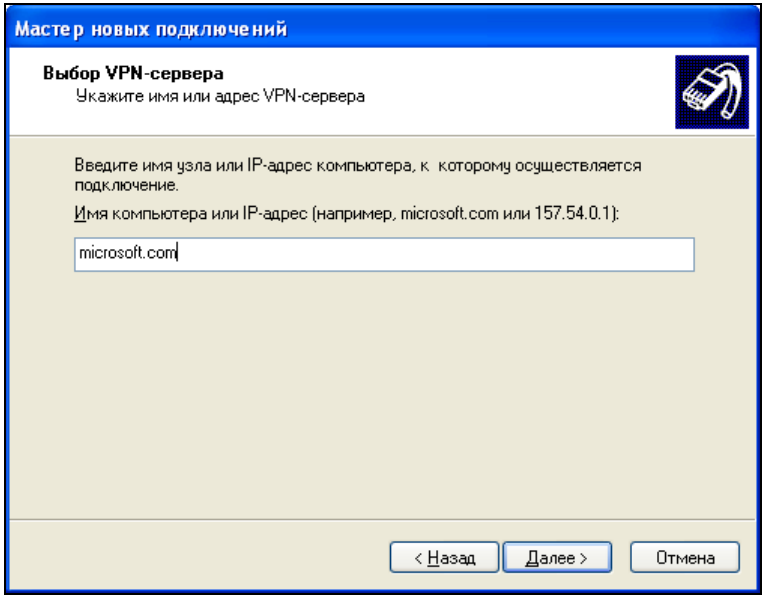


Рис. 35.8. Ввод имени VPN-сервера

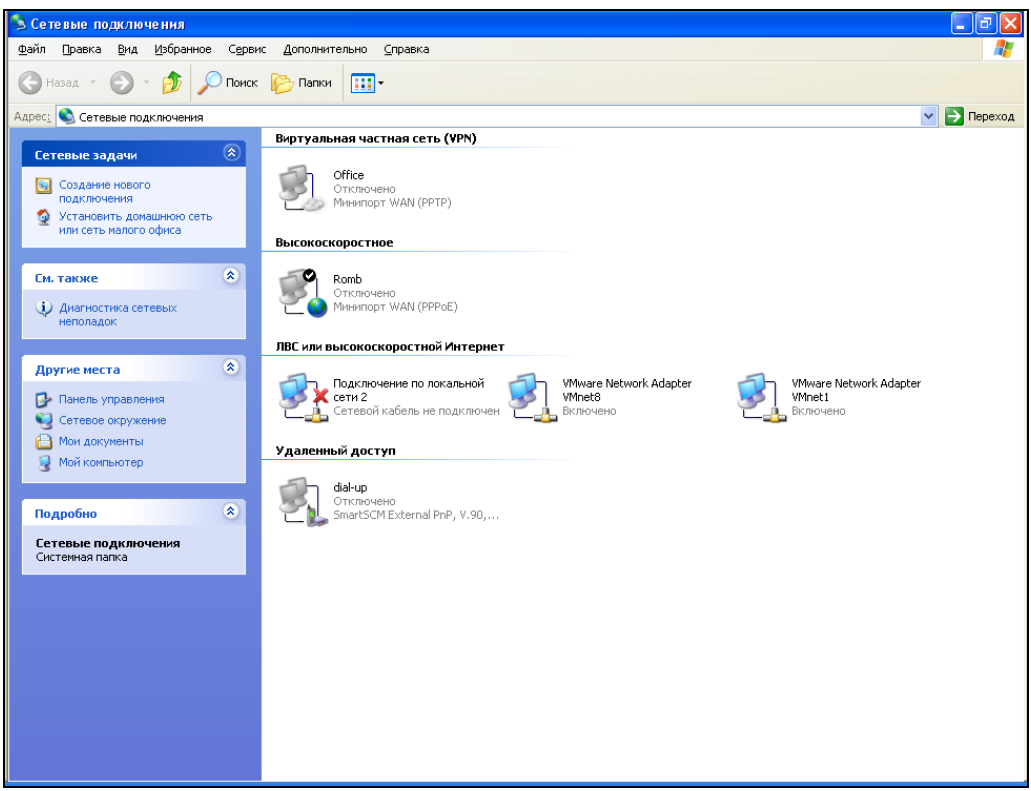


Рис. 35.9. Сетевые подключения

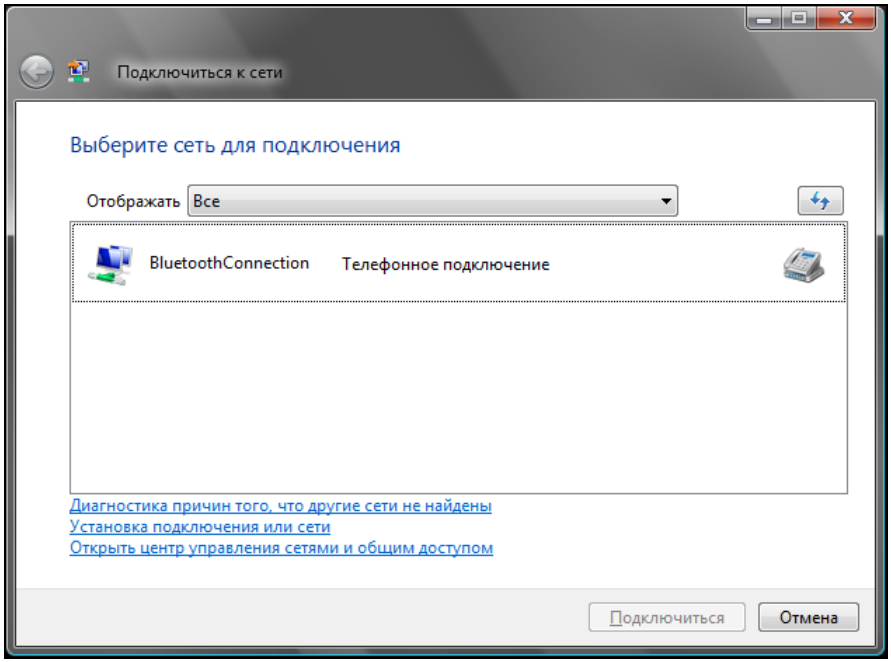


Рис. 35.10. Подключиться к сети

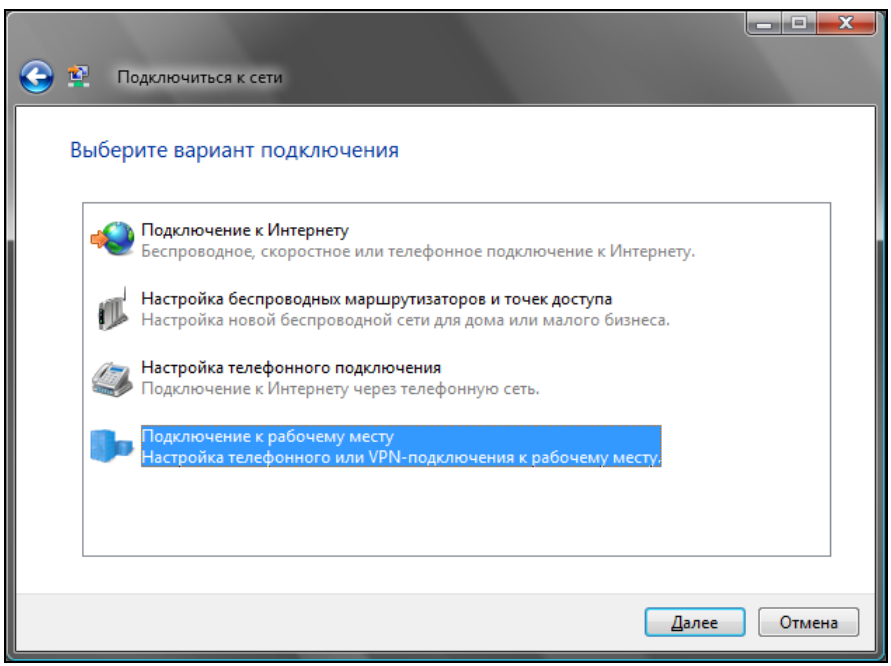


Рис. 35.11. Выбор варианта подключения

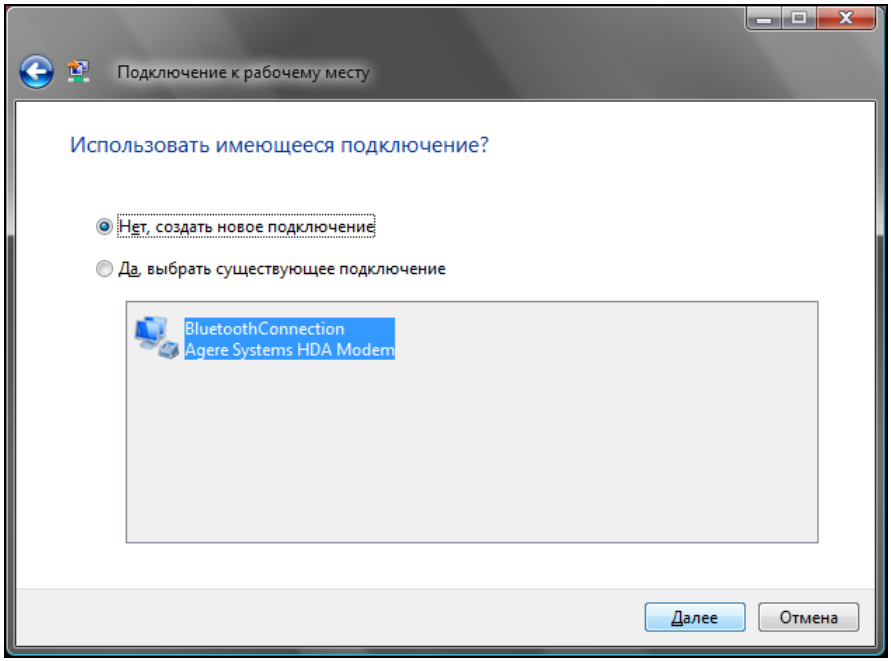


Рис. 35.12. Выбор интернет-соединения

Вам останется лишь ввести параметры подключения: имя VPN-сервера (рис. 35.8), имя пользователя и пароль. Вместо имени VPN-сервера можно ввести его IP-адрес.

После создания VPN-подключения его можно запустить из системной папки **Сетевые подключения** (рис. 35.9).

В Windows Vista/7 VPN-подключение создается аналогично. Выберите команду **Пуск | Подключение**. В открывшемся окне (рис. 35.10) перейдите по ссылке **Установка подключения или сети**. В открывшемся окне (рис. 35.11) выберите **Подключение к рабочему месту.**

Далее процесс настройки ничем не отличается от процесса настройки в Windows XP: нужно выбрать интернет-соединение (рис. 35.12), ввести имя VPN-сервера и т. д.

Глава 36



Защита сетевых сервисов

36.1. Защита Web-сервера

Apache — довольно безопасный сервис, поэтому его защита сводится к установке определенных прав доступа к его конфигурационным файлам. Для начала установим права 700 к каталогу /usr/local/etc/apache22:

```
# chmod 700  /usr/local/etc/apache22
```

После этого никто, кроме вас, не сможет ни просмотреть, ни изменить конфигурационные файлы и файлы протоколов, которые обычно доступны всем желающим для чтения.

Также нужно защитить от изменения конфигурационный файл httpd.conf:

```
# chattr +i /usr/local/etc/apache22/httpd.conf
```

После этого даже вы не сможете изменить данный файл. Если же вам понадобится отредактировать его, снимите атрибут `i`:

```
# chattr -i /usr/local/etc/apache22/httpd.conf
```

Кроме того, желательно в файле httpd.conf установить следующие директивы так:

```
ServerSignature Off
ServerTokens ProductOnly
```

Первая выключает "подпись" сервера на служебных страницах (вывод содержимого каталога, страницы с сообщением об ошибке и т. д.), в которой сообщается версия Apache — чем меньше информации мы предоставляем о нашем сервисе, тем лучше. Вторая выключает вывод версии Apache в HTTP-заголовках — будет выведено только название сервера, но без версии.

В директивах `Directory` *(см. главу 34)* нужно использовать следующие опции:

```
Options -Indexes
Options -Includes
Options -ExecCGI
Options -FollowSymLinks
```

Первая отключает вывод содержимого каталога, если в URL не указан целевой файл, вторая запрещает SSI, третья — запуск CGI, а четвертая — запрещает следовать символическим ссылкам. Хотя иногда проще указать `None`:

```
Options None
```

36.2. Защита FTP

ProFTPD тоже является весьма защищенным сервисом, а его взлом может стать лишь следствием неправильной настройки.

Так, для директивы `DefaultRoot`, задающей корневой каталог сервера, рекомендуется установить значение ~. Как мы знаем, тильда (~) означает домашний каталог пользователя. Следовательно, каждый раз при регистрации пользователя на FTP-сервере корневым каталогом FTP-сервера, который он увидит, станет домашний каталог пользователя. В результате, пользователь не сможет прочитать (а при неправильных правах доступа — изменить) важные системные файлы.

Также рекомендуется включить директиву `RequireValidShell`:

```
RequireValidShell on
```

Если данная директива включена, злоумышленник не сможет установить в качестве оболочки какую-нибудь вредоносную программу. FTP-сервер будет проверять, указана ли программа-оболочка в файле /etc/shells. Если программа не указана в этом файле, то FTP-сервер не позволит ее запустить.

Скрыть вывод версии ProFTPD можно с помощью директивы `ServerIdent`, например:

```
ServerIdent on "My FTP Server"
```

Желательно еще указать директиву `ServerName`:

```
ServerName "Simple FTP Server"
```

36.3. Защита DNS

Серверы DNS обмениваются между собой информацией о зоне. О том, как ограничить передачу зоны, мы говорили в *главе 27*. Но там мы ограничивали передачу зоны по IP-адресу. А что, если злоумышленник каким-то образом подменил целевой DNS-сервер с указанным адресом (например, вывел его со строя и запустил собственный с таким же IP)? Тогда информация о зоне будет передана на сервер злоумышленника.

Чтобы такого не случилось, следует использовать механизм транзакций TSIG (Transaction SIGnatures). Данный механизм предусматривает перед передачей зоны проверку секретного ключа. Если ключ совпадает, информация о зоне будет передана/принята. При несовпадении ключа информация о зоне не будет передана или принята (в случае, если злоумышленник вывел из строя первичный DNS-сервер и пытается передать измененную информацию о зоне на вторичный DNS-сервер).

Первым делом нужно сгенерировать ключ, который затем указать в файле конфигурации каждого сервера. Для этого служит команда:

```
# dnssec-keygen -a hmac-md5 -b 128 -n HOST host1-host2
```

Вам изменять эту команду не стоит. Команда выведет на экран следующую строку:

Khost1-host2.код

Кроме того, в результате выполнения этой команды будут созданы два файла: Khost1-host2.код.key и host2.код.private. Откройте второй файл. В нем будут следующие строки:

```
Private-key-format: v1.2
Algorithm: 157 (HMAC_MD5)
Key: ключ
```

Строку `ключ` нужно скопировать в буфер обмена (или записать на бумаге, если у вас нет графического интерфейса). После этого добавьте следующие строки в файлы named.conf обоих DNS-серверов (первичного и вторичного):

```
key host1-host2 {
  algorithm hmac-md5;
  secret "ключ";
};
```

> ***Внимание!***
>
> Директиву `key` нужно добавить в начало файла конфигурации.

А теперь будьте внимательны и следуйте моим инструкциям.

Откройте файл первичного DNS-сервера и добавьте следующие строки после директивы `key`:

```
server 10.0.0.2  {
key { host1-host2; };
};
```

Затем в директиву `options` добавьте `allow-transfer`, если вы этого еще не сделали:

```
allow-transfer { 10.0.0.2; };
```

Директива `server` указывает, что для обмена зоной с сервером 10.0.0.2 (это IP-адрес вторичного сервера) нужно использовать ключ `host1-host2`.

Теперь откройте файл конфигурации вторичного DNS-сервера. Добавьте следующие строки:

```
server 10.0.0.1  {
key { host1-host2; };
};
```

В директиву `options` добавьте вот такую директиву:

```
allow-transfer { none; };
```

Теперь перезапустите DNS-сервер:

```
# service named restart
```

С серьезной защитой мы справились. Но чуть ранее было показано, как скрыть номер версии Apache и ProFTPD от сетевых сканеров вроде nmap. Проделаем такое же действие для DNS-сервера. В директиву `options` файла named.conf добавьте параметр `version`, что позволит скрыть версию DNS-сервера:

```
options {
        version "";
        …
};
```

36.4. Защита Samba

По умолчанию доступ к серверу Samba предоставляется всем желающим. Это не всегда необходимо. В целях большей безопасности нужно предоставлять доступ только определенным пользователям. Создать пользователей можно с помощью команды `pw`:

```
# pw useradd user1 -c "Samba User" -s /sbin/nologin
# passwd user1
#smbpasswd user1
```

Первая команда добавляет пользователя и устанавливает в качестве его командной оболочки программу /sbin/nologin, которая запускается, возвращает код 0 и завершает работу. Даже если кто-то узнает пароль пользователя, /sbin/nologin не позволит пользователю зарегистрироваться в системе обычным способом.

Вторая команда устанавливает пароль пользователя. Поскольку пользователь не сможет войти в систему, можно назначить любой пароль и даже не запоминать его. А вот третья команда изменяет пароль пользователя, который он должен будет указать при регистрации на сервере Samba. Этот пароль нужно сообщить пользователю.

Затем откройте файл smb.conf и в секции `global` измените параметр `security`:

```
security = user
```



ЧАСТЬ VI

Инструменты системного администратора

Глава 37



Системы мониторинга трафика

Проблема мониторинга трафика волнует каждого администратора. Может быть, сейчас, когда соединения с безлимитным трафиком — реальность, вопрос о мониторинге трафика стоит не так остро. Но ведь нужно же чем-то заняться администратору? BSD-машины нередко настраиваются по принципу "настроил и забыл", у администратора свой принцип: "админ спит, зарплата идет". Но такой образ жизни рано или поздно надоедает, поэтому хочется "прикрутить" что-то полезное к своему серверу.

В этой главе мы рассмотрим две системы мониторинга трафика: достаточно простую систему darkstat, рисующую графики загрузки трафика, напоминающие графики всем известной программы MRTG, и полноценную систему мониторинга NetTAMS. Кому-то хватит первой системы, а у кого-то возникнет необходимость в использовании второй — более сложной и более совершенной.

37.1. Простейшая система мониторинга трафика: darkstat

Установим darkstat:

```
# pkg_add -r darkstat
```

Особой необходимости устанавливать darkstat из портов нет, поэтому устанавливаем ее из пакета, благодаря чему экономим немного времени. Сразу после установки открываем файл /etc/rc.conf и добавляем пока только эти строки:

```
darkstat_enable="YES"
darkstat_interface="em0"
```

Первая строка обеспечивает автоматический запуск darkstat, а вторая — определяет интерфейс, который мы будем мониторить.

Запускаем darkstat:

```
# /usr/local/etc/rc.d/darkstat start
```

Проверяем, запущен ли darkstat:

```
# ps -ax | grep dark
```

Определяем порт, на котором работает darkstat:

```
# sockstat -4 | grep dark
```

```
denhost# pkg_add -r darkstat
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-8.1-release/Lates
t/darkstat.tbz... Done.
denhost# mcedit /etc/rc.conf

denhost# /usr/local/etc/rc.d/darkstat start
Starting darkstat.
denhost# ps -ax ¦ grep dark
 1147  ??  Ss     0:00,06 /usr/local/sbin/darkstat -i em0 --chroot /var/run/dar
 1148  ??  Ss     0:00,01 darkstat: DNS child (darkstat)
denhost# sockstat -4 ¦ grep dark
nobody   darkstat   1147  8  tcp4   *:667                 *:*
denhost#
```

Рис. 37.1. Установка и запуск darkstat



Рис. 37.2. Мониторинг трафика с помощью darkstat

Как видно из рис. 37.1, darkstat запущен и работает на 667 порту. Но обращаться к darkstat пока рано. Нужно сгенерировать немного трафика, чтобы было, что показывать на графике. Интерфейс em0 у меня используется для доступа к Интернету, поэтому я могу сгенерировать трафик путем загрузки какого-нибудь большого файла:

```
# wget http://denix.dkws.org.ua/iso/denix3.iso
```

> ***Примечание***
>
> Программа wget по умолчанию не установлена, для ее установки нужно ввести команду `pkg_add -r wget`.

Вот теперь можно открывать браузер и наслаждаться графиками:

```
http://ip_адрес_узла_где_запущен_darkstat:667/
```

Типичный график darkstat изображен на рис. 37.2.

Но это еще не все. Нужно добавить в файл rc.conf следующую строку:

```
darkstat_flags="--import em0.db --export em0.db"
```

Флаги `--import` и `--export` управляют загрузкой/сохранением статистики в файл/из файла /var/run/darkstat/em0.db при запуске/останове демона darkstat.

Также надо создать сценарий ротации файла em0.db. Код этого файла (назовем его em.sh) представлен в листинге 37.1.

Листинг 37.1. Файл em.sh

```
#!/bin/sh
DB="/var/run/darkstat/em0.db"

getpid() {
PIDFILE="/var/run/darkstat/darkstat.pid"
if [ -f $PIDFILE ]; then
  pid=`cat $PIDFILE`
fi
kill -SIGUSR1 $pid
}

getpid
mv $DB /var/run/darkstat/"darkstat-`date +%Y-%m`"
touch $DB && chmod 666 $DB
getpid
```

Этот файл нужно поместить в каталог /etc/periodic/monthly для его запуска каждый месяц (см. файл /etc/crontab) и сделать его исполняемым:

```
# chmod +x /etc/periodic/monthly/em0.sh
```

На этом все, ваш darkstat полностью готов к работе.

37.2. Система NeTAMS

NeTAMS (Network Traffic Accounting and Monitoring Software) — программа для учета и мониторинга IP-трафика. Поддерживает разные методы сбора статистики, несколько баз данных для хранения информации о трафике (MySQL, PostgreSQL, Oracle, BerkleyDB, Radius), обладает режимом оповещений. Позволяет производить блокировку на базе квот, авторизации, баланса (то есть NeTAMS можно использовать как билинговую систему), дает возможность управлять пропускной полосой, контролировать подмену MAC-адреса, создавать гибкие политики учета и фильтрации трафика. Система довольно серьезная, особенно по сравнению с darkstat, у которой всех этих функций нет.

Перед началом работы с NeTAMS сразу хочу отметить, что вся настройка производилась на базе FreeBSD 8.0. Дело в том, что в файл Makefile системы NeTAMS вкралась ошибка, которую придется исправить вручную. Возможно, в следующей версии FreeBSD или NeTAMS эта ошибка будет исправлена, но пока знайте, что она есть.

Для работы NeTAMS нам понадобятся Apache и MySQL-сервер версии 5.0. Именно 5.0, а не 5.1 или 5.5, потому что с версиями 5.1 и 5.5 NeTAMS почему-то не работает, почему — я не проверял, не было такого желания.

Будем считать, что серверы Apache и MySQL уже установлены и запущены (их настройка рассматривалась в *главе 34*).

Также для работы NeTAMS необходима библиотека libpcap, установим ее:

```
# cd /usr/ports/net/libpcap
# make install clean
```

Теперь установим NeTAMS, но вместо порта, который входит в состав FreeBSD, мы будем использовать порт с сайта разработчиков:

```
# cd /usr/ports/net-mgmt
# fetch http://www.netams.com/files/netams-freebsd-port.tgz
# tar -zxvf netams-freebsd-port.tgz
```

В результате будет создан каталог freebsd-port, переименуем его в netams:

```
# mv freebsd-port netams
```

Настало время исправить ту самую ошибку, о которой я говорил ранее. Откройте файл Makefile:

```
# mcedit /usr/ports/net-mgmt/netams/Makefile
```

Найдите и удалите следующую строку:

```
mysqlclient.16:${PORTSDIR}/databases/mysql51-client \
```

Эта строка должна установить MySQL-клиент версии 5.1, но поскольку NeTAMS не работает с MySQL 5.1, то эта строка лишняя. Также NeTAMS ничего не подозревает об установленном уже MySQL версии 5.0. Если не удалить эту строку, то для разрешения зависимости MySQL 5.0 будет удален, а вместо него будет установлена версия 5.1, но с ней NeTAMS работать не будет. Вот такая ситуация.

Теперь можно собрать порт:

```
# make install clean
```

При сборке зависимого порта GD следует выбрать опцию ICONV. Хотя, скорее всего, библиотека GD с этой опцией у вас уже установлена (она устанавливается при установке PHP).

После сборки добавьте строку автоматического запуска системы в файл /etc/rc.conf:

```
netams_enable="YES"
```

Скопируйте пример файла конфигурации в конфигурационный каталог NeTAMS:

```
# cd /usr/local/etc/netams/
# cp netams.conf.sample netams.conf
```

Пример конфигурационного файла представлен в листинге 37.2. Отредактируйте его по своему усмотрению. Значения, которые вам придется изменить (имена и IP-адреса узлов, порты, пароли и т. д.) выделены полужирным шрифтом.

Листинг 37.2. Пример файла netams.conf

```
# Отключаем отладку
debug none
# Определяем имя пользователя (root) и пароль (1) для доступа к netams
# пользователю root разрешаются все операции (permit all).
# Укажите нормальный пароль! Пароль "1" я использовал, чтобы текст
# поместился в одной строке
user name root real-name Admin password 1 email root@localhost permit all

# настройка сервисов
service server 0
login local
listen 20001
max-conn 6

service processor 0
lookup-delay 60
flow-lifetime 180

# Определяем порты, эти порты будут отображены на графике
# Это весь IP-трафик
policy name ip target proto
# WWW
policy name www target proto tcp port 80 8080 3128 443
# FTP
policy name ftp target proto tcp port 20 21
# POP3
policy name pop3 target proto tcp port 110
# SMTP
policy name smtp target proto tcp port 25
```

```
# SSH
policy name ssh target proto tcp port 22
restrict all drop local pass

# Сортируем порты по группам
unit group name admins acct-policy ip www ftp pop3 smtp ssh
unit group name other_users acct-policy ip www ftp pop3 smtp

# Определяем объекты (юниты), за которыми будем наблюдать:
# тип host — узел
# тип net — сеть
# unit host name comp1 ip A.A.A.A acct-policy ip www ftp pop3 smtp ssh
unit net name LAN ip 192.168.1.0/24 acct-policy ip www ftp pop3 smtp ssh

# Определяем пользователей
unit user name us1 ip 192.168.1.5 parent admin acct-policy ip www ftp pop3
smtp ssh
unit user name us2 ip 192.168.1.7 parent other-users acct-policy ip www ftp
pop3 smtp ssh

# Статистику будем хранить в MySQL, здесь же указываем имя пользователя
# и пароль для доступа к MySQL
service storage 0
type mysql
user root
password пароль_для_mysql
accept all

# Указываем источники данных: откуда будут поступать пакеты
# В моем случае есть только два источника — интерфейсы em0 и em1
service data-source 1
type libpcap
source em0
layer7-detect urls

service data-source 2
type libpcap
source em1
layer7-detect urls

# запускаем сервис мониторинга
service monitor 0
monitor to storage 0
# далее нужно указать наши объекты:
monitor unit comp1
monitor unit LAN
```

```
service alerter 0
report oid 06100 name rep1 type traffic period day detail simple
smtp-server localhost

# определяем, где будем хранить файлы отчетов:
# в каталоге /usr/local/www/netams/stat
service html 0
path /usr/local/www/netams/stat
run 10min
htaccess yes
client-pages all
# адрес веб-сервера
url http://192.168.1.1/netams/
language ru

service scheduler
# обновление отчетов каждые 5 минут
oid 08FFFF time 5min action "html"
```

Конфигурационный файл готов. Запустим NeTAMS:

```
# /usr/local/etc/rc.d/netams start
```

Почти все готово, осталось только настроить Apache. Перейдите в каталог /usr/local/etc/apache22/Includes. В нем должен быть файл netams-apache-freebsd.conf. В этом файле хранятся параметры доступа к каталогам www/netams и www/netams/cgi-bin.

В каталоге /usr/local/www/cgi-bin находятся CGI-скрипты системы NeTAMS, сделаем их исполняемыми:

```
# chmod -R +x /usr/local/www/netams/cgi-bin
```

Отредактируем файл config.cgi:

```
# mcedit /usr/local/www/netams/cgi-bin/config.cgi
```

Найдите в этом файле две строки, содержащие, соответственно, пароль для доступа к SQL и пароль для доступа к NeTAMS:

```
$sql_password="MySQL-пароль"
$sc_passwd="1"              # в конфиге netams указан пароль 1
```

Теперь отредактируем файл admin/config.cgi — в нем надо произвести такие же изменения (изменить переменные `sql_password` и `sc_passwd`):

```
# mcedit /usr/local/www/netams/cgi-bin/admin/config.cgi
```

Перезапустим Apache:

```
# /usr/local/etc/rc.d/apache22 restart
```

Осталось только открыть браузер и зайти в панель управления NeTAMS[1]:

http://ваш_веб_сервер/netams/.

[1] Дополнительная информация (на русском языке) о системе доступна на сайте разработчиков: **http://www.netams.com/doc/index.html**.



Глава 38

Nagios — система мониторинга сети

38.1. Необходимость мониторинга сети

Когда в сети один или два сервера, то достаточно просто понять, какой из них работает, а какой нет. А вот когда в большой сети их несколько десятков, и серверы эти находятся даже в разных городах, то хотелось бы иметь полную картину работоспособности серверов и предоставляемых ими сервисов (FTP, HTTP и т. д.).

Можно написать систему мониторинга на bash — это не так уж и сложно. Впрочем, сложность, конечно, зависит от того, какую информацию вы хотите получить. Если нужно просто определить доступность сервера, то хватит простого сценария, перебирающего в цикле IP-адреса серверов и пингующего эти серверы. Но такой простой сценарий не позволит определить, работают ли запущенные на сервере службы. Придется его усложнять... Но тратить время на разработку сложного сценария не хочется — ведь за нас все уже изобретено. Да и простой сценарий вам разрабатывать тоже не придется — ведь существует сканер портов nmap, позволяющий просканировать всю сеть и определить, доступны ли ее узлы *(см. главу 40)*.

В этой главе будет рассмотрена сложная система мониторинга Nagios[1], позволяющая производить мониторинг сервисов сети (FTP, POP, HTTP, IMAP, SQL и т. п.). Система обладает модульной структурой, позволяющей добавить новые возможности мониторинга, имеется также возможность дописывать собственные модули на bash или Perl. Nagios умеет отправлять уведомления на почту и подавать звуковой сигнал. Можно даже создать Web-страничку, на которой будет отображаться информация о доступности сервисов сети, история сбоев, отчеты о доступности серверов и т. д. Одним словом, Nagios — полноценная система мониторинга.

38.2. Установка Nagios

Для работы Nagios нужен Web-сервер Apache (будем считать, что он уже установлен и запущен, поскольку настройка этого сервера рассматривалась в *главе 34*).

[1] Nagios — далеко не единственная система мониторинга в мире. Еще один превосходный проект — это система Zabbix, познакомиться с которой вы можете по ссылке: **http://www.sergeysl.ru/freebsd-zabbix/**.

Установим Nagios:

```
# cd /usr/ports/net-mgmt/nagios
# make install clean
```

При установке Nagios следует ответить `y` на два вопроса (рис. 38.1):

You need a "nagios" group
Would you like me to create it [YES]? `y`
You need a "nagios" user
Would you like me to create it [YES]? `Y`

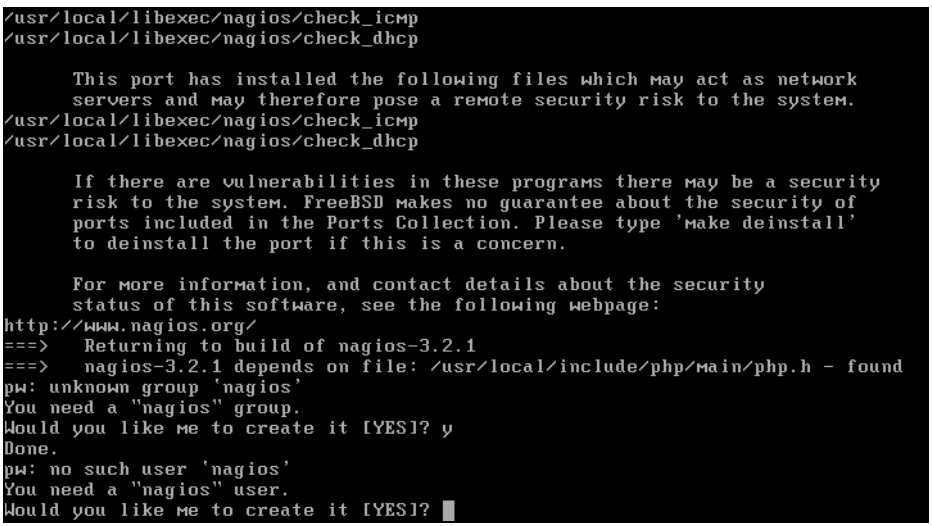


Рис. 38.1. Установка Nagios

Организуем автоматический запуск Nagios, добавив следующую строку в файл /etc/rc.conf:

```
nagios_enable="YES"
```

Теперь откройте файл конфигурации Apache /usr/local/etc/apache22/httpd.conf и найдите в нем строки:

```
<Directory />
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

Сразу после них добавьте следующие строки:

```
ScriptAlias /nagios/cgi-bin /usr/local/www/nagios/cgi-bin/

<Directory "/usr/local/www/nagios">
    Options ExecCGI
    AllowOverride None
```

```
    Order allow,deny
    Allow from all
    AuthName "Nagios Access"
    AuthType Basic
    AuthUserFile /usr/local/etc/nagios/htpasswd.users
    Require valid-user
</Directory>

Alias /nagios /usr/local/www/nagios/

<Directory "/usr/local/www/nagios/cgi-bin">
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
    AuthName "Nagios Access"
    AuthType Basic
    AuthUserFile /usr/local/etc/nagios/htpasswd.users
    Require valid-user
</Directory>
```

Этим вы защитите паролем доступ к каталогу nagios, чтобы никто кроме вас не смог промониторить вашу сеть. Информация о пользователях и паролях хранится в файле /usr/local/etc/nagios/htpasswd.users. У вас еще нет этого файла, создайте его и добавьте в него пользователей:

```
# htpasswd -c /usr/local/etc/nagios/htpasswd.users nagios
# htpasswd  /usr/local/etc/nagios/htpasswd.users admin
```

Перезапустите Apache и обратитесь к Nagios так:

```
http://адрес_сервера/nagios/
```

38.3. Настройка Nagios

Для настройки Nagios перейдите в каталог /usr/local/etc/nagios/. В нем вы найдете только файлы примеров, на основании которых следует создать реальные конфигурационные файлы.

Начнем с файла cgi.cfg-sample — в нем находится конфигурация CGI-части Nagios. Переименуйте этот файл в cgi.cfg. Весь файл со всеми комментариями мы здесь рассматривать не станем, тем более, что много параметров там изменять не придется.

Прежде всего укажите путь к основному конфигурационному файлу:

```
main_config_file=/usr/local/etc/nagios/nagios.cfg
```

Затем укажите путь к файлам на Web-сервере:

```
physical_html_path=/usr/local/www/nagios
```

Часть URL — то, что будет после имени сервера:

```
url_html_path=/nagios
```

Включите аутентификацию:

```
use_authentication=1
```

Определите права для наших пользователей:

```
authorized_for_system_information=nagios,admin
authorized_for_configuration_information=nagios,admin
authorized_for_system_commands=nagios
authorized_for_all_services=nagios,guest,admin
authorized_for_all_hosts=nagios,guest,admin
authorized_for_all_service_commands=nagios
authorized_for_all_host_commands=nagios
```

В конфигурационном файле nagios.cfg прописываются конфигурационные файлы объектов сети, которые нужно мониторить. Например:

```
# Конфигурация для локального (FreeBSD) хоста
cfg_file=/usr/local/etc/nagios/objects/localhost.cfg

# Конфигурация для Windows-машины
cfg_file=/usr/local/etc/nagios/objects/windows.cfg

# Конфигурация для маршрутизатора/коммутатора
cfg_file=/usr/local/etc/nagios/objects/switch.cfg

# Конфигурация для сетевого принтера
cfg_file=/usr/local/etc/nagios/objects/printer.cfg
```

Далее вы выбираете один из файлов (в зависимости от типа объекта), находящихся в каталоге /usr/local/etc/nagios/objects, и на его основе создаете собственный конфигурационный файл. Давайте рассмотрим файл localhost.cfg, подходящий для тестирования компьютера под управлением FreeBSD (листинг 38.1).

Листинг 38.1. Файл localhost.cfg

```
# Определяем тестируемый узел

define host{
# Имя для шаблона — данное имя можно использовать как переменную,
# с помощью которой мы будем ссылаться на этот узел
        use                     freebsd-server
# Имя тестируемого узла
        host_name               localhost
# Псевдоним
        alias                   localhost
# IP-адрес
```

```
        address                 127.0.0.1
        }

# Определим группу хостов, куда поместим все FreeBSD-машины
define hostgroup{
        hostgroup_name  freebsd-servers ; Имя группы
        alias           FreeBSD Servers ; Полное имя группы
        members         localhost       ; Список узлов, входящих
                                        ; в группу, элементы списка
                                        ; разделяются запятой
        }


# Пинг машины

define service{
# Имя сервиса (используется в этом шаблоне)
        use                             local-service
# Имя компьютера, который нужно пропинговать
        host_name                       localhost
# Описание проверки
        service_description             PING
# Команда для проверки
        check_command               check_ping!100.0,20%!500.0,60%
     }

# Определяем сервис, проверяющий свободное место на диске
# для корневого раздела локальной машины. Обычное предупреждение
# вы получите, если останется меньше 20%, а критическое — если
# меньше 10% свободного места на разделе

define service{
        use                             local-service
        host_name                       localhost
        service_description             Root Partition
       check_command               check_local_disk!20%!10%!/
        }

# Определяем сервис, проверяющий количество зарегистрированных
# в данный момент в системе пользователей. Обычно предупреждение —
# если больше 20 пользователей, критическое — если больше 50

define service{
```

```
        use                             local-service
        host_name                       localhost
        service_description             Current Users
       check_command               check_local_users!20!50
        }

# Сервис, проверяющий к-во запущенных процессов в данный момент.
# Обычное предупреждение — если больше 250 процессов, критическое —
# если больше 400

define service{
        use                             local-service
        host_name                       localhost
        service_description             Total Processes
       check_command               check_local_procs!250!400!RSZDT
        }

# Проверяем загрузку локальной машины

define service{
        use                             local-service
        host_name                       localhost
        service_description             Current Load
       check_command               check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
        }

# Проверяем использование свопа на локальной машине
# Обычное предупреждение — если осталось менее 20% свободного свопа
# Критическое — если меньше 10%

define service{
        use                             local-service
        host_name                       localhost
        service_description             Swap Usage
        check_command                    check_local_swap!20!10
        }
# Проверяем доступность SSH-сервиса на локальной машине
# Уведомления по умолчанию отключены, поскольку не всегда
# сервис SSH запущен
define service{
        use                             local-service
        host_name                       localhost
        service_description             SSH
```

```
        check_command                   check_ssh
        notifications_enabled             0
        }

# Проверяем доступность HTTP-сервиса на локальной машине
# Уведомления по умолчанию отключены, поскольку не всегда
# сервис HTTP запущен

define service{
        use                             local-service
        host_name                       localhost
        service_description             HTTP
        check_command                    check_http
        notifications_enabled             0
        }
```

Дополнительные примеры вы найдете в файле objects/templates.cfg-sample. Теоретически, все объекты сети можно описать в одном конфигурационном файле, который потом нужно прописать в файле nagios.cfg, но такая практика не очень удобна. Гораздо удобнее для отдельного объекта создавать отдельный файл.

Вот, собственно, и все. Осталось только запустить Nagios:

```
# /usr/local/etc/rc.d/nagios start
```

Глава 39



Сниффер AimSniff — перехват ICQ-трафика пользователей

39.1. Юридические аспекты

Еще одна глава этой книги может помочь службе безопасности предприятия. Серьезные предприятия довольно часто сталкиваются с проблемой промышленного шпионажа, когда не очень честные сотрудники передают важную информацию конкурентам. Ранее *(см. разд. 29.7)* мы уже разобрались, как определить, какие сайты посещает тот или иной пользователь. А теперь поговорим о перехвате ICQ-трафика. Проанализировав переписку пользователя, можно определить, передает ли пользователь информацию конкурентам или же общается в ICQ сугубо по работе.

Стоит отметить, что перехват ICQ-трафика попадает под действие следующих статей УК РФ:

- 137, часть 2: "Нарушение неприкосновенности частной жизни с использованием служебного положения";
- 138: "Нарушение тайны переписки, телефонных переговоров, почтовых, телеграфных или иных сообщений".

Другими словами, действия, описанные в этой главе, являются уголовным преступлением. Лично я не несу никакой ответственности за изложенную здесь информацию, поскольку публикуется она для общего развития и в образовательных целях — чтобы показать, как работает сниффер AimSniff. Преступите ли вы закон или нет, зависит только от вас — ведь и кухонным ножом можно убить, но вы этого не сделаете?

Можно пойти другим путем и разъяснить пользователям, что все их действия в Сети будут мониториться. Пусть все пользователи также подпишут особое приложение к своему трудовому договору — соглашение о "прослушке". В этом случае с вас снимается ответственность, поскольку пользователь дал на прослушку свое согласие. Но тогда и толку от перехвата трафика не будет — все учтут, что их разговоры контролируются, и будут искать другие средства передачи информации. Теоретически, важную информацию (если объем позволит) можно передать и по SMS — уж мобильный-то телефон вы никак не проконтролируете, для этого нужны соответствующие разрешения.

Еще один способ — вообще запретить использование ICQ и заблокировать ее порты (обычно используется порт 5190) брандмауэром. Тогда пользователи не смогут

передать по ICQ внутреннюю информацию и не будут разговаривать о личном, расходуя драгоценное рабочее время.

В любом случае, что делать в вашей сети — решать только вам. Мое дело маленькое — показать, как использовать AimSniff.

39.2. Установка и настройка сниффера

Установим AimSniff

```
# cd /usr/ports/security/aimsniff
# make install clean
```

После установки сниффера протестируем его:

```
# aimsniff -d=em0 --nodb
```

Опция `-d` задает сетевой интерфейс, на котором будут перехватываться сообщения. Требуется указать локальный интерфейс ("смотрящий" в локальную сеть), следовательно, `em0` в указанной коменде нужно заменить именем вашего интерфейса. Опция `--nodb` не записывает сообщения в базу данных, а просто выводит их на экран — то, что нужно для отладки.

Вместо ICQ-сообщений вы можете увидеть сообщение об ошибке:

Can't locate GDBM_File.pm… (полное сообщение я приводить не стал — оно слишком длинное)

Это сообщение означает, что Perl собран без поддержки GBDM. Пересоберите Perl так:

```
# cd /usr/ports/lang/perl5.8
# make deinstall
# make WITH_GDBM=yes reinstall
```

Но это еще не все. Версия сниффера по умолчанию не умеет читать сообщения на русском в кодировке UTF8. Необходимо скачать и установить уже пропатченную версию сниффера:

```
# fetch http://www.aimsniff.com/releases/aimSniff.Cyr-005.tar.gz
# tar -xf aimSniff.Cyr-005.tar.gz
# chmod +x aimSniff.Cyr-005.pl
# mv aimSniff.Cyr-005.pl /usr/local/bin/asniff
# cd /usr/ports/converters/p5-Unicode-Map8
# make install clean
```

Здесь мы сначала получаем архив со снифером, затем распаковываем его, устанавливаем права и перемещаем в каталог /usr/local/bin. Последние две команды устанавливают порт p5-Unicode-Map8, необходимый для работы пропатченной версии сниффера. Для большего удобства я переименовал снифер в asniff.

Тестируем:

```
# asniff -d=em0 --nodb
```

Осталось настроить снифер для помещения информации в базу данных. Будем считать, что сервер MySQL у вас уже установлен. Сначала нужно создать пользователя, от имени которого будет работать asniff, и базу данных для сообщений ICQ.

Выполните команду:

```
# mysql -u root -p
```

После этого введите три SQL-запроса и команду `exit`:

```
CREATE DATABASE icqmessages;
GRANT ALL ON icqmessages.* TO asniff@localhost IDENTIFIED BY 'asniff';
FLUSH PRIVILEGES;
exit;
```

Мы создали базу данных icqmessages и предоставили полные права по работе с этой базой данных пользователю asniff. Осталось только поместить служебные таблицы в базу данных icqmessages:

```
mysql -u root -p icqmessages \
  < /usr/local/share/doc/aimsniff/table.struct
```

Теперь отредактируйте файл конфигурации /usr/local/etc/aimSniff.cfg (листинг 39.1).

Листинг 39.1. Файл конфигурации /usr/local/etc/aimSniff.cfg

```
dev=em0
filter='tcp and port 5190'
daemon=1
host=localhost
database=icqmessages
user=asniff
password=asniff
useSyslog=1
```

Вам нужно указать только свой интерфейс (в параметре `dev`). Остальные параметры можно не изменять, если вы все делали по книге. Параметр `filter` определяет, какие пакеты нужно фильтровать. В нашем случае мы прослушиваем пакеты, передающиеся по протоколу TCP, порт 5190 (стандартный порт для ICQ). Если вы установили прокси, то укажите порт прокси. Остальные параметры — это параметры доступа к базе данных, измените их в случае необходимости. Параметры `daemon` и `useSysLog` означают, что программа будет работать в режиме демона (в фоновом режиме) и будет использовать SysLog для протоколирования событий.

Просмотреть сообщения, добавленные в таблицу icqmessages, можно или с помощью программы phpMyAdmin (**http://www.phpmyadmin.net/**), или посредством Web-интерфейса для сниффера, скачать который можно по адресу **http://aimsniff.com/releases/was-0.1.2b.tar.gz**. Распакуйте Web-интерфейс в каталог документов Apache и исправьте параметры доступа к базе данных — думаю, вы справитесь с этим и без моих комментариев.



Глава 40

Сканер nmap — программа аудита сети

40.1. Что такое nmap?

Программа nmap предназначена для сканирования как отдельных узлов, так и целых сетей с любым количеством узлов, определения состояния узлов сканируемой сети, а также открытых портов на этих узлах. Сетевой сканер nmap использует для этого множество самых разных методов сканирования[1]: UDP, TCP connect, TCP SYN, ICMP (ping) и т. п.

> ***Примечание***
>
> Сразу хочу отметить, что мы рассмотрим только практическое применение nmap, а теорию (на русском языке) вы и сами сможете прочитать по указанному в сноске адресу (к сожалению, оригинальная страница руководства man на английском) — просто не вижу смысла переписывать страницу руководства, которую можно и так бесплатно получить из Интернета.

Сканер nmap поддерживает множество дополнительных возможностей: определение операционной системы узла, "невидимое" сканирование, вычисление времени задержки, параллельное сканирование, определение неактивных узлов с помощью параллельного ping-опроса, определение наличия брандмауэров, прямое RPC-сканирование, произвольное указание IP-адресов и номеров портов сканируемых сетей.

Результат работы сканера — список отсканированных портов удаленного узла с указанием номера и состояния порта, используемого протокола, а также названия службы, использующей этот порт.

> ***Примечание***
>
> Сканер nmap очень популярен. Он популярен до такой степени, что его можно увидеть даже в фильме "Матрица". В большинстве случаев смотришь фильмы и видишь операционные системы какого-то непонятного происхождения. Создается впечатление, что все эти ОС создавались специально для фильма, чтобы не делать рекламы Microsoft или Apple. А тут старый добрый знакомый — nmap. Не верите? Значит, вы невнимательно смотрели "Матрицу" — вот тот самый эпизод из фильма: **http://www.youtube.com/watch?v=0TJuipCrjZQ**. А вот интервью с создателем nmap: **http://www.xakep.ru/post/20972/default.asp**.

---

[1] Подробно о различиях в методах сканирования можно прочитать в справочной системе (команда `man nmap`) или по адресу: **http://cherepovets-city.ru/insecure/runmap/nmap_manpage-ru.htm**.

40.2. Установка nmap

Сканер nmap — это не нечто мистическое. Это вполне реальная программа, причем абсолютно бесплатная. Она входит в состав портов FreeBSD и многих дистрибутивов Linux. Можно также скачать nmap и с официального сайта (там же вы найдете и Windows-версию): **http://nmap.org/download.html**.

В FreeBSD для установки nmap из портов нужно ввести команды:

```
# cd /usr/ports/security/nmap
# make install clean
# rehash
```

Если же достаточно установить бинарный пакет, воспользуйтесь следующей командой (так будет быстрее):

```
# pkg_add -r nmap
```

40.3. Примеры использования nmap

Познакомимся с основами использования nmap. Все опции мы рассматривать не будем — для этого есть страница руководства, а рассмотрим лишь самые интересные.

Синтаксис вызова nmap такой (запускать nmap нужно с привилегиями root):

```
# nmap параметры цель
```

Здесь `цель` — это узел или список узлов для сканирования.

Предположим, мы хотим знать, какая операционная система запущена на удаленном узле. Для этого нужно запустить nmap с опцией `-O`:

```
# nmap -O узел
```

Вот результат сканирования узла с запущенной Ubuntu Linux 10.04:

Starting Nmap 5.21 (nmap.org) at 2010-08-23 10:28 EST
Nmap scan report for 192.168.1.1
Host is up (0.0040s latency).
Not shown: 999 closed ports
PORT STATE SERVICE
22/tcp open ssh
MAC Address: XX:XX:XX:XX:XX:XX
Device type: general purpose
Running: Linux 2.6.X
OS details: Linux 2.6.19 — 2.6.32
Network Distance: 1 hop

OS detection performed. Please report any incorrect results at nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.20 seconds

Как можно видеть, узел работает под управлением Linux с ядром 2.6.19 — 2.6.32 (как раз, в Ubuntu 10.04 ядро 2.6.32). Также видно, что открыт один порт — 22 (ssh), все остальные порты закрыты.

Пояснение

У порта может быть три состояния: открыт, фильтруемый, нефильтруемый. Первое состояние означает, что удаленная машина прослушивает данный порт. Второе — что брандмауэр блокирует доступ к этому порту и nmap не может определить его состояние. Третье состояние означает, что порт просто закрыт.

Если nmap запущен для сканирования узла локальной сети, то он также сообщает и MAC-адрес узла (свой MAC-адрес я скрыл).

Сканер позволяет просканировать сразу несколько узлов. Для этого их адреса нужно указать так:

```
nmap 192.168.1.1-100
```

В приведенном примере будут просканированы узлы от 192.168.1.1 до 192.168.1.100. Можно также указать имена узлов через пробел, например, так:

```
nmap host1 host2
```

Чтобы просто просканировать узел на предмет открытых портов, укажите просто его адрес, никаких опций указывать не нужно, например:

```
nmap 192.168.1.2
```

Вывод будет примерно таким:

Interesting ports on den (192.168.1.2):

Not shown: 1712 closed ports

PORT STATE SERVICE

22/tcp open ssh

80/tcp open http

Nmap done: 1 IP address (1 host up) scanned in 0.240 seconds

Как уже отмечалось ранее, nmap позволяет также узнать список запущенных служб — для этого нужно использовать опцию `-sV`:

```
nmap -sV 192.168.1.2
```

Вот вывод этой команды:

Starting Nmap 5.21 (nmap.org) at 2010-08-23 10:45 EST

Nmap scan report for den (192.168.1.2)

Host is up (0.099s latency).

Not shown: 962 closed ports, 32 filtered ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 4.7p1 Debian 8ubuntu1.2 (protocol 2.0)

80/tcp open http Apache httpd 2.2.8 ((Ubuntu) PHP/5.2.4-2ubuntu5.10 with Suhosin-Patch)

Service Info: OS: Linux

Service detection performed. Please report any incorrect results at nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 10.51 seconds

Еще один интересный тип сканирования — кто онлайн? Сканер позволяет просканировать сеть и определить доступные узлы. Для этого служит опция `-sP`:

```
nmap -sP 192.168.1.1-254
```

Вы увидите краткий отчет — активен ли тот или иной узел сети и MAC-адрес сетевого адаптера узла, если узел активен:

Starting Nmap 5.21 (http://nmap.org) at 2010-10-07 11:11 EST

Nmap scan report for 192.168.1.1
Host is up (0.0011s latency).
MAC Address: xx:xx:xx:xx:xx:xx (Intel Corporate)
Nmap scan report for 192.168.1.4
Host is up (0.018s latency).
MAC Address: xx:xx:xx:xx:xx:xx (Intel Corporate)
Nmap scan report for 192.168.1.5
Host is up (0.0020s latency).
MAC Address: xx:xx:xx:xx:xx:xx (Intel Corporate)
...

Если узел закрыт брандмауэром, вы увидите сообщение, что узел или выключен, или закрыт брандмауэром. Но nmap все же позволяет получить хоть какую-то информацию об этом узле, а именно: включен ли узел. Для этого служит опция `-PN`. По крайней мере, вам станет ясно: узел или выключен, или включен, но закрыт брандмауэром.

На этом все. Теперь, когда вы знаете, что существует сканер nmap, для вас не составит особого труда ознакомиться со страницей его руководства.

> ***Примечание***
>
> Нужно отметить, что администраторы не любят, когда несанкционированно сканируют их узлы. Во-первых, это неэтично. Во-вторых, у вас могут возникнуть проблемы — все зависит от организации, которую вы сканируете. Само сканирование не противозаконно, но все мы понимаем, что просто так никто ничего не сканирует. Если вам хочется поэкспериментировать, лучше использовать тестовый сервер nmap: **scanme.nmap.org**.



Глава 41

Антивирусная проверка трафика

41.1. Постановка задачи

В этой главе будет рассмотрена антивирусная проверка Web-трафика, проходящего через прокси-сервер Squid. А нужна ли такая проверка? С одной стороны — нет, поскольку на каждом клиентском компьютере должен быть установлен антивирус. Антивирус нужен, даже если шлюз обеспечивает проверку трафика — ведь вирус может попасть на компьютер не только из Всемирной паутины. Есть множество других способов распространения вируса: по почте, через сменные носители (флешки, дискеты), по ICQ и т. д.

С другой стороны, дополнительная защита — это в любом случае хорошо. Кроме того, мы сможем заставить наш шлюз хоть немного заняться полезной деятельностью и использовать свои вычислительные возможности с большей отдачей. А то смотришь на статистику процессора шлюза и видишь, что в лучшем случае он загружен процентов на 10, а обычно и того меньше. По сути, процессор работает вхолостую, а ведь мы платили деньги за те самые 2 гигагерца, рассчитывая, что они нам понадобятся. Поэтому нужно хоть как-то использовать их по назначению. И антивирусная проверка трафика как нельзя лучше подходит для этого дела.

Как уже было сказано ранее, мы будем проверять трафик, проходящий через прокси-сервер Squid. А это означает, что Squid должен быть установлен и настроен *(см. главу 29)*.

Существует несколько способов "прикрутить" антивирус к Squid: c-icap, редиректоры и антивирусный прокси. C-icap — это реализация ICAP-сервера. Прокси-серверы вроде Squid используют протокол ICAP для фильтрации трафика в связке прокси + ICAP + антивирус. Но этот вариант отпал, так как настроить его не удалось, а устраивать пляски с бубном не имело смысла, поскольку имеются и более простые в реализации способы. Редиректоры я использовал для настройки аналогичной функции в Linux[1], поэтому захотелось чего-то новенького. И было принято решение настраивать связку Squid и антивируса через антивирусный прокси. В портах был найден отличный вариант для этого — HAVP (HTTP AntiVirus Proxy). Есть и еще одна причина выбора этого варианта — в качестве антивируса

[1] Интересующиеся могут прочитать об этом в других моих книгах — где рассматривается настройка прокси-сервера Squid в Linux.

я решил привлечь бесплатный антивирус ClamAV (уж так не хотелось покупать коммерческий антивирус!). Пусть ClamAV по некоторым критериям и не дотягивает до коммерческих собратьев, зато он, как уже отмечено, бесплатен и, в отличие от некоторых других бесплатных проектов, постоянно развивается и регулярно обновляется. Так вот — HAVP обладает уже встроенной поддержкой ClamAV, и ничего не придется допиливать напильником. В итоге вся настройка сводится к установке HAVP, ClamAV и редактированию двух конфигурационных файлов.

Перед тем как приступить к настройке, опишу свои впечатления от эксплуатации данной связки. Все работает не хуже, чем при настройке через редиректоры, может быть даже и лучше. Но не в этом суть. Хотел загрузить процессор, чтобы он не простаивал, а получилось, что нагрузил память — HAVP более требователен к оперативной памяти, чем к частоте процессора. Поэтому чем больше на сервере оперативной памяти, тем лучше. Но в любом случае поставленная цель была достигнута. Я получил стабильную нагрузку на процессор и оперативную память — теперь "железо" работает, а не простаивает. Кстати, о необходимом объеме оперативной памяти. В небольшой сети на 30 компьютеров, из которых одновременно в Интернете по каналу 2 Мбит/с работают не более 20-ти, загрузка оперативной памяти составила 200–300 Мбайт. Так что в идеале на вашем шлюзе должно быть установлено 512 Мбайт оперативной памяти (понимаю, что есть еще подкачка, но использование подкачки, сами понимаете, тормозит систему).

41.2. Установка HAVP и ClamAV

Как обычно, устанавливаем HAVP из портов:

```
# cd /usr/ports/www/havp/
# make install clean
```

Включаем опции `SSL` и `CLAMAV` (рис. 41.1).

Ну, а ClamAV, как уже отмечалось ранее, устанавливать отдельно не нужно, поскольку при установке HAVP будет также автоматически установлен и ClamAV.

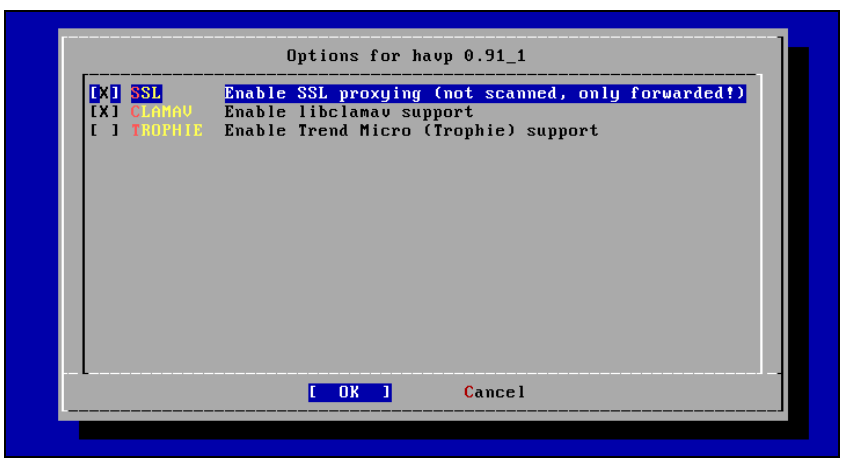


Рис. 41.1. Установка HAVP

41.3. Настройка ClamAV и HAVP

Первым делом настроим ClamAV. Откройте файл /usr/local/etc/clamd.conf. Большинство параметров можно оставить как есть. Можно было бы вообще не редактировать этот файл, но я рекомендую установить следующие опции так:

```
LogTime yes
ScanMail no
MaxFileSize 10M
MaxFiles 1500
```

Первая опция включает протоколирование времени. Вторая — отключает проверку почты (ведь мы будем проверять только HTTP-трафик). Третья — устанавливает максимальный размер файла. По большому счету, даже 10 Мбайт — это перебор, вирусы вряд ли будут распространяться в таких больших файлах. Обычно файл с вирусом небольшой, чтобы вирус мог максимально быстро проникнуть на компьютер жертвы. Так что если на вашем шлюзе оперативной памяти немного, можно уменьшить это значение до 3 Мбайт (в моем случае оперативную память девать некуда — 2 Гбайт, поэтому я и установил здесь 10 Мбайт). Четвертая директива — максимальное количество файлов, проверенное за одну сессию. Вряд ли загруженная страничка потянет за собой больше 1500 файлов (картинок, скриптов, роликов и т. д.). Вы должны понимать, что мы будем использовать ClamAV для проверки Web-трафика (а не для проверки файловой системы локального компьютера), поэтому 1500 файлов — и так слишком много.

Теперь откройте конфигурационный файл /usr/local/etc/freshclam.conf, содержащий параметры обновления ClamAV, и закомментируйте следующую опцию:

```
#NotifyClamd /usr/local/etc/clamd.conf
```

Дело тут в том, что по умолчанию после обновления баз данных антивируса утилита `freshclam` должна уведомить об этом демон clamd. Однако нам он не нужен, и мы даже не будем его запускать. А чтобы утилита `freshclam` каждый раз после обновления антивирусных баз не ругалась, что не может связаться с clamd, мы и закомментировали эту опцию.

Настроив ClamAV, займемся настройкой HAVP. Большинство опций файла его конфигурации /usr/local/etc/havp/havp.config можно оставить по умолчанию, рассмотрим здесь только несколько наиболее полезных:

```
LOG_OKS false
FORWARDED_IP true
PORT 3127
TEMPLATEPATH /usr/local/etc/havp/templates/ru
ENABLECLAMLIB true
CLAMDBDIR /var/db/clamav
CLAMMAXFILESIZE 10
CLAMMAXRECURSION 3
```

Первую опцию на время отладки полезно установить в `true`, но затем, когда вы убедитесь, что все нормально работает, установите ее обратно в `false`. Вторая оп-

ция нужна, чтобы в протокол был занесен IP-адрес пользователя, скачавшего страничку с вирусом, а не значение 127.0.0.1. Опция `PORT` задает порт нашего прокси. Если у вас другой порт, измените его (часто используют порт номер 8080).

Опция `TEMPLATEPATH` задает путь к шаблонам HAVP — заготовкам страничек, которые будут показаны в случае обнаружения вируса. Мы используем путь, отличный от пути по умолчанию, — каталог /usr/local/etc/havp/templates/ru. В него следует поместить оригинальные шаблоны из каталога /usr/local/share/examples/havp (они на английском языке) и перевести их на русский. Думаю, с этим вы справитесь сами. А если такого желания нет, можно использовать и оригинальные английские шаблоны, только не забудьте скопировать их в каталог /usr/local/etc/havp/templates/ru.

Следующая директива включает библиотеку для "сопряжения" с ClamAV — обязательно установите эту директиву в `true`. Далее следуют три опции, относящиеся к ClamAV — путь к антивирусной базе, максимальный размер файла, максимальный уровень рекурсии (при вложенной проверке Web-страницы).

Обновим антивирусные базы ClamAV:

```
# /usr/local/etc/rc.d/clamav-freshclam start
```

Starting clamav_freshclam.

Запустим HAVP:

```
# /usr/local/etc/rc.d/havp start
```

Starting havp.

Starting HAVP Version: 0.91

Проверим, работает ли HAVP:

```
# netstat -an |grep 3127
```

tcp4 0 0 127.0.0.1.3127 *.* LISTEN

Итак, HAVP запущен и работает, а ClamAV обновлен. Обеспечим теперь автоматическое обновление ClamAV и запуск HAVP. Для этого в файл /etc/rc.conf добавьте строки:

```
clamav_freshclam_enable="YES"
havp_enable="YES"
```

41.4. Настройка Squid

Осталось только настроить Squid, чтобы он перенаправлял весь трафик на HAVP. Откройте конфигурационный файл /usr/local/etc/squid/squid.conf и добавьте в него строки:

```
cache_peer 127.0.0.1  parent    3127  0     default no-query
acl FTP proto FTP
always_direct allow FTP
always_direct allow SSL_Ports

acl no_avir urlpath_regex -i \.avi$ \.mp3$
always_direct allow no_avir
never_direct allow all
```

Первая строка ключевая — она-то и обеспечивает отправку трафика на родительский прокси — на наш HAVP. Далее мы разрешаем передачу трафика по протоколам FTP и SSL без проверки антивирусом. FTP-трафик проверять нам и не нужно, а SSL проверять нет смысла — он зашифрован. Кстати, и аська (ICQ) без прямого редиректа SSL-портов работать не будет.

Последние три строчки запрещают проверку AVI- и MP3-файлов. В них по идее вирусов быть не должно, да и размер такаих файлов обычно не маленький, поэтому, чтобы не захламлять оперативную память, проверять их не будем.

Перезапустите Squid, перейдите на любой другой компьютер сети и попробуйте загрузить следующие тест-файлы (это не вирусы, но любой антивирус должен на них среагировать), чтобы убедиться, что все работает:

- **http://www.eicar.org/download/eicar.com.txt**;
- **http://www.eicar.org/download/eicar_com.zip**;
- **http://www.eicar.org/download/eicarcom2.zip**.



Глава 42

SMS-рассылка

42.1. Постановка задачи

Рано или поздно в крупной компании возникнет необходимость в рассылке SMS своим сотрудникам или клиентам. Например, интернет-провайдер за день до окончания пакета может отправлять своим клиентам напоминание о необходимости пополнить счет.

Организовать рассылку SMS средствами FreeBSD довольно-таки просто, что в очередной раз подчеркивает универсальность этой операционной системы. Можно, конечно, использовать одну из уже готовых Windows-программ для рассылки SMS-ок. Однако весь смысл в запуске программы рассылки на сервере, обслуживающим клиентов провайдера, — ведь только он знает, когда у того или иного пользователя заканчивается срок пакета.

Для реализации задуманного нам понадобится какой-нибудь недорогой мобильный телефон (а зачем дорогой?), компьютер с установленной FreeBSD и USB-кабель для подключения телефона к компьютеру. Предпочтительнее использовать именно USB-кабель, потому что он позволяет подзаряжать аккумулятор телефона. Теоретически, можно вообще не подзаряжать телефон — он будет получать питание по USB. Правда, не все телефоны могут заряжаться по USB, например, злосчастный (3 ремонта за 8 месяцев, каждый ремонт по 2 недели) Phillips 960 заряжаться по USB не может.

42.2. Установка SMS Tools

Для рассылки сообщений мы будем использовать SMS Tools, поэтому первым делом установим следующий порт:

```
# cd /usr/ports/comms/smstools3
# make install clean
```

Подключаем телефон и вводим команду:

```
# kldstat -v | grep 'u[pl]*com'
```

Если вы не увидели следующих строк (или подобных им):

202 ucom
203 uhub/uplcom

то вам придется перекомпилировать ядро *(см. главу 21)* и включить вот эти опции:

```
device          ucom
device          uplcom
```

Надеюсь, что заветные строки в выводе команды `kldstat` будут, и вам не придется перекомпилировать ядро.

Теперь редактируем файл /usr/local/etc/smsd.conf. Пример этого файла представлен в листинге 42.1.

Листинг 42.1. Файл /usr/local/etc/smsd.conf

```
# Используемое устройство
devices = MOBILE0
# Журнал демона smsd
logfile = /var/log/smsd.log
# Уровень протоколирования
loglevel = 5

# Работа от root не очень хороша из соображений безопасности
# но мы будем надеяться, что код smsd не будет использовать
# наш сервер в своих целях
USER = root
GROUP = root

# Не изменяйте эти параметры
PIDFILE= /var/run/smsd.pid
INFOFILE= /var/run/smsd.working

# Теперь прописываем телефоны. Можно прописать несколько телефонов
# Активный телефон указывается в параметре devices
# Можно даже одновременно использовать два или больше телефонов —
# тогда smsd отправит SMS через первый доступный телефон. Если
# желаете использовать несколько телефонов, пропишите их идентификаторы
# в параметре devices через запятую, например:
# devices = MOBILE0,MOBILE1,MOBILE2
[MOBILE0]
# порт
device = /dev/cuaU0
# скорость порта
baudrate = 115200
# отключаем аппаратное управление потоком
rtscts = no
# строка инициализации телефона
```

```
# возможно, ее придется отредактировать
init = AT+CPMS="ME","ME","ME"
init2 = AT+CNMI=1,1,0,2
# указываем приоритет входящих сообщений
incoming = high
# включаем отчет о доставке
report = yes
# задержка перед отправкой в секундах
send_delay = 15
```

Почти все. В файл /etc/rc.conf добавим следующую строку для автоматического запуска демона smsd:

```
smsd_enable="YES"
```

Теперь запустим сам smsd:

```
# /usr/local/etc/rc.d/smsd start
```

Если вы все сделали правильно, можете начать отправлять SMS-ки командой:

```
sendsms 7xxxxxxxxxx 'Your balance: 0 USD'
```

Номер телефона нужно указывать в международном формате, но без "+".

Через некоторое время на указанный телефон должна прийти SMS-ка.

42.3. Русификация SMS

Текст сообщения приходится набирать на английском (можно в транслите). Ежели гордость не позволяет отправлять нерусские SMS-ки, придется в файле /usr/local/bin создать сценарий smsconvert, представленный в листинге 42.2.

Листинг 42.2. Сценарий smsconvert

```
#!/bin/sh

FILE=`mktemp /tmp/sms_tmp_XXXXXX`

if [ ! `grep '[А-Яа-я]' $1 > /dev/null` -o `grep 'Alphabet:\s*U' $1 >
/dev/null` ]
    then exit 0
fi

cat $1 | awk '{if(NF==0) {c=1} if(c==0 && NF>0 && $0!~/Alphabet:[
\t]*U/){print}}' > $FILE
echo Alphabet: Unicode >> $FILE
```

```
cat $1 | awk '{if(NF==0) {c=1} if(c==1){print}}' | iconv -t UCS-2BE >> $FILE

mv $FILE $1
chmod 664 $1
```

Подобный этому сценарий имеется в каталоге /usr/local/share/smstools. Можете за основу взять любой из них. Теперь установите права запуска для этого сценария:

```
# chmod + smsconvert
```

И укажите путь к сценарию в файле smsd.conf:

```
devices = MOBILE0
checkhandler=/usr/local/bin/smsconvert
```

Вот теперь, действительно, все. Осталось только написать сценарий, отправляющий каждому пользователю его баланс (в помощь вам *глава 12*). Такой сценарий нужно добавить в расписание `cron` и запускать каждый день (только не слишком рано!).



Глава 43

Шифрование разделов

43.1. Необходимость в шифровании

FreeBSD является довольно защищенной системой. Но ни пароли, ни средства принудительного контроля доступа MAC (Mandatory Access Control) не защитят ваши данные, если злоумышленник получит физический доступ к вашему серверу. Достаточно извлечь жесткий диск и подключить его к своему компьютеру. Дальнейшее — дело техники.

Совсем по-другому будет выглядеть ситуация, если данные на диске зашифрованы. Тогда злоумышленнику придется очень постараться, чтобы их расшифровать. В большинстве случаев он потратит столько времени, что информация на жестком диске окажется неактуальной.

Существуют разные системы шифрования. Некоторые из них предлагают шифровать отдельные файлы. Нам такие системы совершенно не подходят, поскольку они довольно громоздкие и сильно снижают производительность компьютера. В FreeBSD имеются технологии, шифрующие целые разделы жесткого диска, причем шифрование осуществляется прозрачно и на лету, так что в открытом виде данные на жесткий диск никогда не попадают — записываются уже зашифрованные данные. Аналогично, расшифровка данных тоже происходит прозрачно — программы даже не подозревают, что данные на диске были зашифрованы.

В FreeBSD можно использовать технологию gdbe (GEOM Based Disk Encryption), позволяющую зашифровать диск на уровне модульной дисковой структуры GEOM (шифруется каждый сектор диска 128-битным ключом с использованием алгоритма AES), и криптографическую файловую систему geli (использует алгоритмы шифрования AES, Blowfish и 3DES). Мы рассмотрим здесь и gdbe, и geli.

43.2. Технология gdbe

43.2.1. Включение gdbe

Первым делом нужно перекомпилировать ядро с включением поддержки gdbe:

```
options GEOM_BDE
```

Можно обойтись и без перекомпиляции ядра, добавив модуль `geom_bde`:

```
# kldload geom_bde
```

Если загрузка модуля прошла успешно (не было сообщений об ошибках), тогда в файл /boot/loader.conf добавьте строку:

```
geom_bde_load="YES"
```

43.2.2. Шифрование нового жесткого диска

Сейчас мы добавим винчестер, на котором создадим единственный раздел. Данные этого раздела будут зашифрованы. Позже вы сможете подмонтировать этот раздел к каталогу /var, чтобы базы данных и данные Web-сервера были зашифрованы. А пока мы будем монтировать этот раздел к каталогу /crypted.

В *главе 14* было показано, как подключить новый жесткий диск, — следуйте указаниям из этой главы. Так что будем считать, что у нас имеется SATA-винчестер с именем /dev/ad6, на нем создан один раздел /dev/ad6s1, и весь слайс адресует раздел /dev/ad6s1c.

Создадим каталог для блокировок gdbe и каталог /crypted:

```
# mkdir /crypted
# mkdir /etc/gdbe
```

В каталоге /etc/gdbe будут находиться файлы блокировок, необходимые gdbe для доступа к зашифрованному разделу. Без файла блокировки gdbe не сможет расшифровать данные. Для каждого зашифрованного раздела создается отдельный файл блокировки.

Перед началом работы с gdbe-разделом его нужно инициализировать. Инициализация производится только один раз. При инициализации задается имя файла блокировки (/etc/gbde/ad6s1c.lock):

```
# gbde init /dev/ad6s1c -i -L /etc/gbde/ad6s1c.lock
```

Программа gdbe запустит редактор (надеюсь, вы уже установили редактором по умолчанию программу ее?), в котором вы должны установить размер сектора в 2048 (для файловых систем UFS и UFS2):

```
# Sector size is the smallest unit of data which can be read or written.
# Making it too small decreases performance and decreases available space.
# Making it too large may prevent filesystems from working.  512 is the
# minimum and always safe.  For UFS, use the fragment size
#
sector_size     =       2048
[...]
```

Далее gdbe запросит пароль, который предназначен использоваться для защиты данных. Пароль будет запрошен дважды, в обоих случаях его следует ввести одинаковым. Старайтесь придумать сложный и длинный пароль, поскольку надежность шифрования прямо пропорциональна сложности введенного вами пароля. Пароль должен содержать символы разного регистра ("большие" и "маленькие") и цифры.

В процессе выполнения команды будет создан файл блокировки. Чтобы файлы блокировки корректно распознавались стартовым сценарием /etc/rc.d/gbde, их имена

должны заканчиваться строкой ".lock", иначе придется модифицировать файл /etc/rc.d/gbde, чего делать не хочется.

> ***Внимание!***
>
> Делайте резервные копии файлов блокировок! Если вы потеряете файл блокировки, то расшифровать данные будет очень сложно. Лично я не знаю, как это сделать, да и расшифровка данных с потерянными файлами блокировок не поддерживается программой gdbe. Разработчики gdbe тоже не предоставляют подобных услуг, так что придется или смириться с потерей данных, или искать специалистов, специализирующихся на расшифровке информации, но никто не даст вам полной гарантии, что данные получится расшифровать.

После инициализации устройства нужно подключить его к системе:

```
# gbde attach /dev/ad6s1c -l /etc/gbde/ad6s1c.lock
```

Просмотрим, подключилось ли устройство:

```
# ls /dev/ad6*
```

/dev/ad6 /dev/ad6s1c.bde
/dev/ad6s1
/dev/ad6s1c

Если вы увидели в выводе команды файл-устройство /dev/ad6s1c.bde, значит, устройство успешно подключено к системе. Самое время создать на нем файловую систему:

```
# newfs -U -O2 /dev/ad6s1c.bde
```

После создания файловой системы мы можем подмонтировать устройство к каталогу /crypted:

```
# mount /dev/ad6s1c.bde /crypted
```

На этом все. Можно приступить к использованию устройства.

43.2.3. Монтирование уже зашифрованного жесткого диска

Разберемся, как смонтировать зашифрованный жесткий диск после перезагрузки. Первым делом следует присоединить устройство к системе, указав файл блокировки:

```
# gbde attach /dev/ad6s1c -l /etc/gbde/ad6s1c.lock
```

Затем нужно проверить файловую систему:

```
# fsck -p -t ffs /dev/ad6s1c.bde
```

После этого можно смонтировать устройство:

```
# mount /dev/ad6s1c.bde /crypted
```

Размонтировать устройство можно так:

```
# umount /crypted
# gbde detach /dev/ad6s1c
```

> ***Внимание!***
>
> Конфигуратор sysinstall до сих пор не совместим с gdbe, поэтому перед его запуском для каких-либо настроек нужно обязатеьно размонтировать gdbe-устройство.

43.2.4. Автоматическое монтирование gdbe-устройств

Для автоматического монтирования gdbe-устройств нужно добавить в файл /etc/rc.conf следующие строки:

```
gbde_autoattach_all="YES"
gbde_devices="ad6s1c"
gbde_lockdir="/etc/gbde"
```

Как вы уже догадались, вторая строка — это список зашифрованных устройств, а третья — это каталог с блокировками.

При подключении устройства будет запрошен пароль, созданный при инициализации устройства. Другими словами, при загрузке, помимо пароля пользователя, придется ввести пароль и для каждого зашифрованного устройства. Не очень удобно, но такова цена безопасности данных.

43.3. Криптографическая файловая система geli

43.3.1. Особенности geli

Криптографическая файловая система geli отличается от технологии шифрования разделов gdbe — она предоставляет другие возможности и использует другую схему шифрования. Принципиальные отличия следующие:

- при наличии возможности аппаратной криптографии, geli будет использовать ее (в отличие от gdbe);
- использование криптографических сервисов ядра (см. `man crypto`);
- поддержка алгоритмов AES, Blowfish и 3DES;
- высокая производительность — благодаря простому шифрованию сектор-сектор;
- поддержка двух независимых ключей шифрования;
- поддержка шифрования файловых систем случайным одноразовым ключом (используется для разделов подкачки и временных файловых систем).

Какой алгоритм шифрования лучше использовать? AES появился позже (то есть якобы является более совершенным), чем Blowfish, зато Blowfish, по оценкам независимых экспертов, быстрее работает, да и максимальная длина ключа у Blowfish больше — 448 битов против 256 у AES. Не используйте только 3DES — это самый слабый из поддерживаемых алгоритмов.

43.3.2. Включение поддержки geli

Как и в случае с gdbe, поддержку geli можно включить либо путем перекомпиляции ядра, либо с помощью модуля. В первом случае в файл конфигурации ядра нужно добавить строки:

```
options GEOM_ELI
device crypto
```

Во втором случае добавьте следующую строку в файл /boot/loader.conf:

```
geom_eli_load="YES"
```

Затем надо перезагрузить систему.

43.3.3. Шифрование с помощью geli

Прежде всего нужно сгенерировать ключевой файл. Сгенерированный ключевой файл послужит частью главного ключа для шифрованного провайдера, который мы подмонтируем в каталог /crypted. С использованием содержимого ключевого файла создается набор случайных данных, которым и будет зашифрован главный ключ. Он также будет защищен паролем. Размер сектора в нашем случае равен 4096 байтам. Такой размер сектора является оптимальным по соотношению производительность/использование дискового пространства.

Для создания ключевого файла мы применим устройство /dev/random, являющееся генератором случайных чисел:

```
# dd if=/dev/random of=/root/ad6.key bs=64 count=1
# geli init -s 4096 -K /root/ad6.key /dev/ad6
```

Enter new passphrase:

Reenter new passphrase:

Первая команда создает ключевой файл, а вторая инициализирует geli с использованием созданного ключевого файла. В нашем случае /dev/ad6 — это имя устройства (второй SATA-винчестер).

Далее нужно связать наш ключ с провайдером (с устройством /dev/ad6):

```
# geli attach -k /root/ad6.key /dev/ad6
```

Enter passphrase:

В результате будет создано устройство /dev/ad6.eli:

```
# ls /dev/ad6*
```

/dev/ad6 /dev/ad6.eli

Осталось только создать файловую систему на устройстве /dev/ad6.eli и смонтировать ее к каталогу /crypted:

```
# dd if=/dev/random of=/dev/ad6.eli bs=1m
# newfs /dev/ad6.eli
# mount /dev/ad6.eli /crypted
```

После этого вы можете использовать зашифрованную файловую систему. Размонтировать ее можно так:

```
# umount /crypted
# geli detach ad6.eli
```

43.3.4. Автоматическое подключение geli-устройств

Для автоматического подключения geli-устройства добавьте в файл /etc/rc.conf следующие строки:

```
geli_devices="ad6"
geli_ad6_flags="-p -k /root/ad6.key"
```



ЧАСТЬ VII

Теория и практика системного администрирования



Глава 44

Стратегия администрирования

В седьмой, заключительной, части книги мы поговорим о суровых буднях системного администратора. С одной стороны, мы будем говорить о практике, а с другой стороны, все эти разговоры — чистой воды теория. Потому что все равно у вас будет все иначе, чем написано в книге.

Вы можете быть самым профессиональным администратором в мире, у вас может быть лучшая команда (из таких же профессионалов, как вы), но без четкой стратегии администрирования вы далеко не зайдете.

Инструкции, приведенные в этой главе, одинаково пригодятся как единственному администратору в компании, работающему на полставки, так и руководителю IT-отдела.

44.1. Структура IT-службы

Внутри любой IT-службы можно выделить следующие подразделения:

- **руководство** — сюда входят руководитель отдела ("верховный" администратор) и несколько его заместителей. Как показывает практика, это подразделение не должно быть большим. Одного–трех человек вполне достаточно (но не более 5% от всего IT-отдела);
- **отдел администрирования** — к этому подразделению относятся как раз сами администраторы, выполняющие всю основную работу по поддержанию всей сети в рабочем состоянии. Размер отдела, наверное, самый большой, хотя многое зависит от профиля самой компании;
- **отдел разработки** — создается по мере необходимости. Одни компании занимаются разработкой программного обеспечения, другим нужны программисты для доработки уже существующих программных продуктов. Например, если компания занимается разработкой программного обеспечения, то отдел разработки будет самым большим. А вот в некоторых организациях, наоборот, отдела разработки может и не быть. Экономия…
- **отдел поддержки** — занимается либо поддержкой собственных пользователей, либо поддержкой клиентов (если компания занимается разработкой программного обеспечения). Сотрудники этого отдела также должны заниматься обучением работе с собственным программным обеспечением.

Подобная структура может существовать в компании среднего или крупного размера. Иногда весь IT-отдел состоит из двух-трех человек: руководителя (главный администратор) и двух помощников. Руководитель заменяет все подразделение управления, а помощники выполняют функции администрирования и поддержки пользователей. Отдела разработки в таких компаниях, как правило, нет: или один из администраторов выполняет некоторые функции программиста (например, "допиливает" 1С), или же компания нанимает сторонних специалистов (для разовой работы так выходит намного дешевле).

Когда ваш отдел разрастается, помните о золотом правиле: один руководитель на десять человек. Как только число человек превысит 10, нужно разделяться. Например, 5 человек будут администрировать сеть, а 5 — заниматься поддержкой пользователей. Вы же — будете управлять всеми ими. Поверьте, это не просто. В идеале один человек может управлять группой не более 10 человек. Если в группе окажется больше 10 человек, нужно разделить ее на несколько групп и в каждой группе назначить главного. Тогда вы будете управлять всего несколькими руководителями групп, а они уже — своими группами.

44.2. И руководство, и пользователи довольны. Миф или реальность?

Часто бывает так, что либо руководство, либо пользователи недовольны. Самый сложный случай, когда недовольны и руководство, и пользователи. Например, руководство недовольно сроками внедрения новой системы управления предприятием, но вы задерживаетесь с ее внедрением, поскольку не успеваете параллельно заниматься поддержкой пользователей. А текущие проблемы (у кого-то зависает компьютер, "упал" Интернет, нужно заменить картридж в принтере) занимают, действительно, много времени. Хуже всего, когда выполнение одних обязанностей мешает вам исполнять другие. В итоге руководство будет недовольно вдвойне: вы не выполнили план и на вас (на ваш отдел) есть жалобы пользователей.

Давайте разберемся, как сделать так, чтобы все были довольны. С руководством, с одной стороны, проще — нужно выполнять их указания и вовремя предоставлять отчеты о проделанной работе. А как же пользователи? А пользователь доволен в следующих случаях:

- когда его компьютер работает без сбоев и зависаний;
- когда "работает Интернет", и не просто работает, а быстро работает;
- когда установлено программное обеспечение, с которым умеет работать сотрудник (желательно, чтобы оно работало без сбоев);
- когда есть приветливая служба поддержки.

Казалось бы, все логично и просто. Но вот компьютер пользователя начал зависать и самопроизвольно перезагружаться. Проблема с оперативной памятью налицо. В компьютере установлено два модуля оперативной памяти. Вы один извлекли. В результате компьютер стал работать нормально, но медленнее. Теперь пользователь жалуется, что компьютер работает медленно. Где взять новый модуль?

Правильно, в магазине. Но не покупать же его за свои средства? Приходится обращаться непосредственно к начальству (либо к финансовому директору, бухгалтеру и т. д.), объяснять проблему и получать "добро" на покупку нового модуля.

Но часто бывает так. Директор спрашивает, работает ли компьютер в данный момент? Вы отвечаете, что работает, но медленно. Но, услышав слово "работает", остальное он слышать не хочет. Денег нет, экономия, кризис и т. д. Это с одной стороны. А с другой стороны — пользователь, которому все равно, как вы сделаете, чтобы компьютер работал быстрее. Он ведь должен жаловаться именно вам. Получается, что администратор как бы между двух огней. С одной стороны — руководство и экономия, с другой — пользователи. К этому нужно привыкать и пытаться находить компромиссы. Чему учиться в книгах по психологии, а не здесь...

С Интернетом тоже часто бывают проблемы. Вспоминаю свой опыт работы "верховным" системным администратором. Мой рабочий день начинался в 9 часов утра, собственно, раньше добраться до места работы не получалось. Зато некоторые пользователи, в том числе и финансовый директор, любили приходить пораньше. А Интернета "нету". И дело не в настройках сервера. А просто по утрам он "падал" — такой был провайдер. К моему приходу на работу меня уже поджидали недовольные пользователи. Звонок в службу поддержки провайдера, полчаса ожидания, соединение установлено. Хорошо, что в число "любителей Интернета" входил финансовый директор. Закончилось все сменой провайдера, и договор с новым провайдером подписывал именно он.

Идем дальше — программное обеспечение. Понимаю, что знания пользователей — это их личные проблемы. Вы не обязаны обучать каждого пользователя компьютерной грамотности. Ведь есть же специальные курсы. Но бывает так, что в целях экономии компании переходят на открытое программное обеспечение. Например, вместо привычного пользователям пакета MS Office устанавливается бесплатный OpenOffice.org. Он похож на MS Office, но как бы ни была хороша копия — это не оригинал. А еще хуже, когда устанавливается Linux. Нет, Linux — отличная операционная система, но уж больно она отличается от привычной Windows. Некоторого программного обеспечения для Linux нет, поэтому запускать его придется в эмуляторе wine. Не исключено, что под эмулятором программы будут работать не так, как нам бы этого хотелось.

Экономия экономией, но о пользователях тоже надо заботиться. Какой прок от бесплатной ОС и от экономии, если первые полдня пользователь не может выполнить привычные действия, а другую половину дня администратор объясняет ему, как работать в Linux (да, вместо того, чтобы заниматься более полезными делами)? Поэтому переход на Linux нужно начинать постепенно. В первую очередь Linux следует устанавливать на компьютеры, где не требуется специальное программное обеспечение. Если на компьютере установлен браузер, почтовый клиент и пакет MS Office (типичный компьютер типичного менеджера) — это первый кандидат для перехода на Linux. Желательно также учитывать уровень квалификации самого пользователя. Если перед вами женщина предпенсионного возраста, то на ее компьютер Linux лучше установить в последнюю очередь. Linux не так уж и сложен, просто к нему придется привыкать. А постепенный переход означает, что вы будете получать 5 жалоб в день, а не 100.

С FreeBSD в некотором роде проще. Как правило, вы даже не станете пытаться установить BSD на рабочие станции пользователей, а будете использовать BSD на собственном сервере, следовательно, жаловаться вам, если что-то пойдет не так, останется только самому себе. Но такие жалобы разрешаются гораздо проще.

Разберемся, что не нравится пользователям (это тоже важно учитывать):

- простои (даже запланированные);
- поврежденные или нечаянно удаленные файлы;
- обновления (пользователям не нравятся изменения, особенно если они вносят некоторые неудобства в привычный рабочий процесс);
- долгие объяснения, почему система (сеть) не работает.

Вообще пользователю не нравится, когда система (сеть) не работает. Им хочется, чтобы система работала круглосуточно, а почему она работает (или не работает) — они знать не желают. Да, проблемы — ваши. Всем интересен результат, а обеспечить этот результат должны вы. И вообще, как бы обидно это ни звучало, пользователям вообще все равно — существуете вы или нет. Они вспоминают об администраторе только тогда, когда что-то ломается.

А о простоях нужно предупреждать пользователей заранее, например, за день — чтобы они успели закончить текущие дела. Хотя все равно найдутся недовольные, так что особенно можете не обращать на них внимание. Есть такая категория людей, которая всем и всегда недовольны.

Относительно удаленных файлов — тут решать вам: или вы сообщаете пользователям, что они и только они отвечают за создание резервных копий своих собственных файлов, или вам же придется заниматься резервных копированием самостоятельно. Когда пользователи вынуждены сами делать резервные копии — гора с плеч (с ваших), но вы — плохой администратор. А если "бэкапы" делаете вы, то вы — администратор хороший. Но забот у вас станет намного больше, чем вы думаете. И придется подыскать хороший программный комплект для резервного копирования по сети (не будете же вы вручную скачивать данные с каждого компьютера?). Я рекомендую пакет Amanda Network Backup (**http://www.amanda.org**) — он бесплатный и может использоваться как в Linux, так и в Windows. Также можно использовать программу Clonezilla (**http://clonezilla.org/**), но она больше подходит для создания образа только что установленной ОС, что позволит быстро развернуть образ в случае сбоя, да и клонировать компьютеры с помощью Clonezilla проще.

Итак, будем считать, что и руководство, и пользователи — довольны. Но что это мы только о них и говорим? А чего бы хотелось самим администраторам? Примерно вот этого:

- имеются все необходимые для выполнения обязанностей ресурсы (в том числе и финансовые, и неограниченный доступ к Интернету);
- имеется нормальный парк "железа", а не музейные экспонаты, год выпуска которых можно установить только путем экспертизы;
- чтобы не приходилось отвлекаться от выполнения первоочередных обязанностей (нужно донастроить брандмауэр iptables, а тут пришел пользователь, который не знает, на какую кнопку нажать);

- чтобы можно было творчески подойти к решению задачи без лишнего контроля и придирок руководства;
- чтобы количество рабочих часов было нормированным (ненормированный рабочий день нравится далеко не всем — есть свои интересы помимо работы).

Но на практике, к сожалению, бывает наоборот. Старые компьютеры, ограниченный доступ к Интернету (с учетом трафика даже для администратора — и такое проходили), множество недовольных пользователей (компьютеры старые, постоянно зависают и медленно работают). Так что нужно быть готовым к разным ситуациям.

44.3. Роль главного администратора

Этот раздел будет полезен руководителю IT-отдела или главному администратору. Вот основные обязанности руководителя:

- определение направления деятельности, предоставление необходимых ресурсов (да, "выбивать" денежку от вышестоящего начальства придется именно вам);
- набор и увольнение персонала;
- отчеты перед вышестоящим начальством;
- распределение задач, контроль их выполнения;
- решение конфликтов между сотрудниками;
- контроль развития сети (чтобы она не потеряла масштабируемости);
- ведение локальной документации (схема сети, расположение компьютеров, установленное программное обеспечение и т. д.).

Может, вам показалось, что в этом списке кое-чего не хватает? Действительно, может, чего-то и не хватает — все зависит от специфики организации, в которой работает администратор. Одно могу сказать точно: руководитель IT-отдела не должен напрямую общаться с пользователями. Решением их проблем пусть занимаются другие администраторы. Однако пользователи могут жаловаться главному администратору на действия (или бездействие) других администраторов, например, нахамил, игнорирует просьбы и т. д.

Особого внимания заслуживают набор и увольнение персонала — от этого зависит, с кем вы будете работать. Управление персоналом — нелегкая задача. Ведь нужно оценивать не только технические знания, но и личностные качества. Человек может быть отличным специалистом в своей области, но как человек… Продолжать не буду, иначе на обложке появится надпись "Осторожно! Ненормативная лексика". Если вы все-таки хотите взять такого человека в отдел, то явно не в службу поддержки пользователей. Пусть настраивает серверы или выполняет ту работу, которую лучше него никто не сможет сделать. Однако, если есть возможность, стоит все же вообще отказаться от приема такого сотрудника в отдел.

Технические знания сотрудника оценить довольно легко (при условии, конечно, что вы сам — профессионал своего дела). Несколько коварных вопросов, и вы будете знать уровень профессиональных знаний будущего сотрудника. Если у вас есть время и вам не лень, можно разработать даже систему тестирования знаний.

Совсем не обязательно разрабатывать программный комплекс, достаточно создать в текстовом редакторе что-то наподобие анкеты, распечатать ее, и пусть кандидаты в сотрудники отвечают на вопросы этой анкеты. А вы уже оцените уровень знаний.

Надо, тем не менее, понимать, что анкетирование дает только относительное представление о знаниях и способностях сотрудника. Например, вы задаете вопрос, на который человек не знает правильного ответа, поскольку никогда с этим не сталкивался. Но если человек способен к самообучению, то он найдет информацию в Интернете и сможет решить поставленную задачу. Увы, это можно узнать только на практике. А если каждому предоставлять доступ к Интернету во время заполнения анкеты, то все будут специалистами высочайшего класса. Но как раз для этого и существует испытательный срок (обычно 3 месяца). За этот срок сотрудник проявит себя и как специалист, и как человек. Вам важно определить, вписывается ли сотрудник в коллектив, как он общается с пользователями, клиентами и другими администраторами. Если человек не вписался в коллектив, то его лучше уволить, даже если с профессиональной стороны к нему претензий нет. Если его оставить, то головная боль в будущем вам обеспечена — вы только и будете заниматься разрешением конфликтов внутри коллектива.

Есть два способа сформировать команду профессионалов. Первый заключается в том, что вы нанимаете уже опытных администраторов. Второй — вы обучаете имеющихся сотрудников со средним уровнем квалификации. С одной стороны, опытный администратор сразу включается в работу. С другой, у опытных администраторов есть один недостаток — их приходится от многого отучать. Так происходит со всеми профессионалами своего дела, а не только с администраторами. Например, водитель-дальнобойщик, попав за руль "легковушки", при переключении передач делает "перегазовку", а этого не только не нужно делать на "легковушке", но и нельзя, так как вредно для сцепления. Опытные администраторы также сразу требуют привилегированного доступа ко всему, но им его сразу давать нельзя, поскольку вы еще ничего о них не знаете — можно ли им доверять или нет.

Когда же вы выращиваете администраторов в своем коллективе, то всегда можете обучить их так, как вам нужно. Но на все это нужно время, которого иногда мало.

Кроме управления персоналом вам придется заниматься распределением заданий и контролировать их выполнение. При этом избегайте следующего:

- не мешайте своим сотрудникам — нужно управлять задачами, но не вмешиваться. Если вы постоянно стоите "над головой" у младших администраторов, это будет им только мешать;
- недоразумений при выполнении заданий — иногда задания не выполняются, потому что человек, который должен выполнить то или иное задание, думает, что его должен выполнить кто-то другой. Каждый сотрудник должен знать свое задание и сроки его выполнения;
- излишнего расходования ресурсов — иногда руководители, чтобы быть точно уверенными в выполнении задания, поручают одно и то же задание нескольким сотрудникам или отделам (чтобы каждый сотрудник или каждый отдел лично работал над заданием). Такая тактика в некоторых случаях себя оправдывает, но не забывайте, что действия сотрудников нужно кому-то координировать. Например,

вы поручили Иванову и Петрову настроить Apache. Иванов внес одни изменения в конфигурационный файл, а Петров, ничего не подозревающий о действия Иванова, через час внес в этот файл другие изменения. В итоге сервер так и не работает. Поэтому, если предположить, что Иванов с заданием справится, лучше поручить Петрову другое задание — так будет рациональнее. Это как ехать на двух машинах в булочную только из-за перестраховки, что одна из машин может поломаться. Целесообразнее выбрать "самую исправную" машину и отправить ее за хлебом. Шутка...

Руководителю IT-отдела постоянно приходится общаться с вышестоящим руководством. Именно от него можно получить средства, необходимые для выполнения работы (на покупку расходных материалов, комплектующих, программного обеспечения). Отчеты о проделанной работе тоже нужно составлять для начальства.

Говоря с начальством, помните, что оно не имеет никакого представления о том, чем занимаются системные администраторы. Поэтому старайтесь говорить (составлять отчеты) максимально понятно. Старайтесь избегать профессионального сленга и использования непонятных начальству аббревиатур, иначе вы не убедите их в необходимости покупки нового оборудования и найма дополнительных сотрудников.

Когда пишете отчет для начальства, придерживайтесь пяти простых правил:

- все должно быть максимально понятным;
- старайтесь не использовать непонятных (для начальства!) терминов, обязательно поясняйте, что означает тот или иной термин;
- расшифровывайте аббревиатуры (также обязателен перевод расшифровки аббревиатуры на русский язык);
- когда требуете средства на новое оборудование, обязательно поясняйте, что даст замена старого оборудования новым;
- когда требуете расширение штата, обязательно объясните, чем будет заниматься каждый новый сотрудник.

Надеюсь, рекомендации, приведенные в этой главе, хоть как-то помогут вам. А в следующей главе мы поговорим об уходе за аппаратными средствами.



Глава 45

Уход за "железом"

45.1. Обязанности администратора

В обязанности администратора очень часто входит не только настройка программного обеспечения, но и уход за "железом". Да, чистка всего компьютерного парка. И если внешнюю чистку можно возложить на плечи самих пользователей (ну не будете же вы всем мониторы протирать), то внутреннюю чистку должны производить только вы сами.

Правда, чистка — это не единственное занятие администратора. Вам также придется заниматься ремонтом (в пределах разумного) и модернизацией компьютеров предприятия.

Некоторые предприятия, отличающиеся особо большим парком компьютеров, нанимают сторонние фирмы для чистки и профилактики компьютеров. Если вы работаете на таком предприятии — вам повезло. Вы будете заниматься только настройкой программного обеспечения. Настроили Linux-серверы и забыли, останется только время от времени Windows на рабочих станциях переустанавливать. Однако в большинстве случаев на таких услугах экономят, и все действия по уходу за компьютерами (всеми компьютерами, а не только серверами) должен выполнять системный администратор.

Теоретически, парк до 30 компьютеров в состоянии обслуживать один человек. А вот если компьютеров больше, то нужен как минимум еще один человек — помощник. Поэтому если вы оказались на предприятии, где больше 30 компьютеров и всего один администратор (то есть вы), смело требуйте расширения штата ИТ-отдела. На первый взгляд кажется, что и 50 компьютеров вы "потянете". Но это не так. Поверьте мне на слово. Обслуживать даже 30 компьютеров — это довольно тяжело. Представьте, что ночью была гроза, а утром вы обнаружили, что 5 компьютеров (да, грозозащита — это хорошо, но современные реалии таковы, что она присутствует далеко не всегда) не работают. По закону подлости — это компьютеры директора, главного бухгалтера, ведущего менеджера и других немаловажных лиц на предприятии. Какой компьютер нужно отремонтировать первым? Куда бежать, если сгорел блок питания? В общем, приятного мало. А вот когда у вас есть помощники, проблема решится быстрее.

Есть и другая ситуация, которая намного проще предыдущей. Предположим, что вам нужно проложить сеть в другой корпус предприятия. Эту операцию намного

проще выполнять с помощником, нежели самому. Точнее, без помощника вам вообще не обойтись.

Подобных примеров могу привести очень много, но толку от них не будет. Лучше приступим к формированию собственного обменного фонда.

45.2. "Про запас", или обменный фонд

Как администратору, вам нужно сформировать собственный обменный фонд аппаратных средств и периферийных устройств. Некоторые комплектующие и периферийные устройства имеют "привычку" выходить из строя с завидной регулярностью. Пример: дешевые китайские мыши, клавиатуры. Вы донастраиваете сервер, а у главного бухгалтера поломалась мышь. В итоге вам нужно все бросить и ехать в ближайший компьютерный магазин, вместо того, чтобы спокойно доделать текущую работу. А все это время и нервы. А если у вас будет запасная мышка, то проблема решится за пару минут.

Кроме клавиатур и мышек в обменном фонде должны быть:

- **жесткие диски** — обычно хватит одного-двух запасных SATA-дисков (они наиболее распространены на современных компьютерах). Если в вашем компьютерном парке есть компьютеры с устаревшими IDE-дисками, запасайтесь и ими. Сейчас достать жесткий диск (новый) с IDE-интерфейсом довольно сложно — в наличии они бывают не всегда, а под заказ приходится ждать 1–2 дня. Полагаю, через год найти IDE-диск будет еще сложнее. И тогда в случае выхода из строя IDE-диска придется покупать новую материнскую плату (скорее всего, процессор и оперативную память тоже придется менять) или специальный SATA-контроллер для подключения нового SATA-диска. Ну, в самом деле, не выбрасывать же компьютер из-за одного жесткого диска? Понимаю, что он устарел, но для работы с текстом (офисный компьютер — это не игровая станция) он вполне подходит.

  Если найти новый IDE-диск вы не можете, попробуйте купить б/у. Вот только перед покупкой его желательно проверить на наличие плохих секторов (в Linux для этого используется программа badblocks. Жаль что в FreeBSD нет подобной програмы). Много покупать б/у-дисков не стоит — это коты в мешке — сегодня работают, а завтра — "посыпались". Так что уже сегодня можете "порадовать" руководство, что вам нужно минимум 2 IDE-диска (если они, конечно, используются на ваших компьютерах) и 1–2 SATA-диска. SATA-диски всегда есть в наличии, да и учитывая, что они более "молодые", вероятность выхода их из строя ниже, чем у IDE-дисков. Поэтому одного-двух вполне хватит;
- **блоки питания ATX** — никогда не угадаешь, когда блок питания выйдет из строя. Поэтому нужно запастись парочкой блоков питания. Не покупайте дешевые блоки мощностью 250 Вт. Для современного компьютера 300 Вт — это необходимый минимум, а лучше покупать блоки питания мощностью 350 Вт. Иначе "голодание" вашему компьютеру обеспечено. Запасные блоки питания также пригодятся вам при чистке основных блоков питания. Чистить блок питания рекомендуется раз в год, но процесс чистки занимает минут 20–30, и, чтобы

сервер не простаивал, вы можете установить запасной блок питания, а основной блок питания почистить без всякой спешки;

- **картриджи для принтеров** — заниматься самостоятельной заправкой тонера нельзя — это вредно для здоровья. Лучше отнести картридж в сервисный центр. На заправку картриджа потребуется 1–2 дня (учитывая загрузку сервисного центра), а чтобы в это время принтер не простаивал, нужен запасной картридж. Хорошо, если все принтеры на предприятии одного производителя и одной модели. В противном случае придется покупать несколько разных картриджей. Не спешите с покупкой картриджей — нужно выяснить, какие принтеры используются чаще всего, а потом уже планировать закупку картриджей для этих принтеров;
- **носители информации** — у системного администратора просто должен быть запас DVD-болванок и флешек. Вот только раздавать болванки и флешки всем пользователям не рекомендую — иначе они очень быстро израсходуются. Используйте эти носители для своих целей, например, для резервного копирования;
- **привод DVD-RW** — в приводе DVD (как и CD) много механических частей, а это означает, что его вероятность выхода из строя довольно высока. Лучше всего купить USB-привод. Его можно использовать как временную подмену вышедшему из строя приводу как обычного компьютера, так и ноутбука;
- **коммутатор** (switch) — иногда эти "коробочки" ломаются (или сгорают отдельные порты). Чтобы привести сеть в чувство, требуется замена коммутатора. Хорошо, если запасной коммутатор есть под рукой. Нужно покупать коммутатор как минимум на 16 портов, а еще лучше — на 24 порта. Чем больше компьютеров смогут работать после выхода из строя коммутатора, тем лучше.

Вот вроде бы и все. Переходим к чистке компьютеров.

45.3. Чистка компьютеров. Профилактика системы охлаждения

Чистку компьютера следует производить хотя бы раз в полгода, а чистку блока питания — раз в год. Для чистки компьютера нужно обзавестись обычным пылесосом. Можно даже использовать компактный автомобильный пылесос. Стоит такой пылесос недорого, зато он очень удобен.

Думаю, понятно, что чистка должна быть сухой. Никаких влажных салфеток и моющих средств! При чистке компьютера особое внимание уделите радиатору процессора. Возможно, придется снять процессор, чтобы хорошо очистить радиатор. Радиатор, забитый пылью, отнюдь не способствует отводу тепла, а выполняет обратную функцию — еще больше нагревает процессор.

Если вентилятор процессора или блока питания издает посторонние звуки (помимо естественного шума), его нужно смазать. Для смазки подойдет любое машинное масло — хватит одной-двух капель масла. Но это только в том случае, если вентиляторы оснащены специальным отверстием для смазки. Оно находится под наклейкой по центру вентилятора. Обычно на вентиляторах блока питания такое отверстие имеется, а вот что касается вентилятора процессора, то тут ваши шансы

найти такое отверстие равны 50%. На одних вентиляторах такое отверстие есть, на других — нет. Иногда из-за этого приходится менять вентилятор — он шумит, а смазать его нельзя.

Со стационарными компьютерами все очень просто. А вот с ноутбуками все намного сложнее. Чтобы добраться до вентилятора процессора, приходится полностью разбирать некоторые модели ноутбуков. Вот пример разборки ноутбука HP: **http://smanuals.ru/electronics-repair/hp-compaq-6730s-6735s-disassembly-replace-cooling-fan.html**. Без этого руководства я бы никогда не добрался до вентилятора процессора. А если бы и добрался, то наверняка что-нибудь сломал. Отсюда вывод — перед разборкой ноутбука попробуйте найти соответствующее руководство в Интернете. Если же такого руководства нет, а заднюю крышку все равно снять не получается, лучше обратитесь в сервисный центр. В противном случае ремонт выйдет дороже, чем обычная чистка.

Кстати, уборку в серверном помещении я бы не доверял уборщицам. А то у нас часто бывает, что любая уборщица хуже лучшего хакера. Одно неаккуратное движение шваброй, и сервер "умер". Кто будет убирать в вашем серверном помещении — решайте сами. Но убирать там нужно регулярно.

45.4. Охлаждение компьютеров

Нормальная температура для работы компьютеров — примерно 20 °C. Но сами понимаете, что такие условия встречаются далеко не всегда, особенно летом. Сейчас программа SpeedFan показывает температуру ядра процессора 71 °C (ноутбук HP 6735s), температура в помещении около 30 °C, пора включать кондиционер.

А вот теперь начинается самое интересное. Многие из нас привыкли подбирать кондиционер по площади помещения, в котором он будет установлен. Но это в корне не правильно. Нужно рассчитать общую тепловую нагрузку, а затем подбирать кондиционер, соответствующий этому параметру.

Тепловая мощность измеряется в БТЕ (BTU, British thermal unit, британская термическая единица). 1 Вт примерно равен 3.412 БТЕ/час. Пусть в помещении находятся 10 компьютеров, каждый из которых потребляет по 400 Вт. Рассчитаем тепловую мощность:

$$10 \times 400 \times 3.412 = 13648 \text{ БТЕ/час}$$

Кроме компьютеров источниками тепла являются сами пользователи и осветительные приборы. Пусть в помещении включено 5 лампочек по 100 Вт каждая, рассчитаем тепловую мощность:

$$5 \times 100 \times 3.412 = 1706 \text{ БТЕ/час}$$

Один пользователь выделяет тепла на 300 БТЕ/час. Выходит, наши 10 пользователей создадут нагрузку в 3000 БТЕ/час.

Осталось учесть тепловую нагрузку от окон, стен и потолка. Например, если у вас солнечная сторона, то нагрузка от окон будет больше, чем на противоположной стороне здания. Также если вы находитесь на последнем этаже, то нагрузка будет еще и от крыши, которая летом постоянно нагревается. Все это пусть считает

специалист по установке кондиционеров. Вы же добавите к полученному показателю свои значения. Осталось сравнить общую тепловую нагрузку с эффективностью охлаждения кондиционера (параметр EER), подробно об этом можно прочитать тут: **http://en.wikipedia.org/wiki/Energy_efficiency_ratio**.

Некоторые компании экономят на охлаждении, устанавливая кондиционеры только в кабинетах директоров, а "рабочий класс" обречен спасаться от жары с помощью вентиляторов. В этом случае ничем компьютеру особо не поможешь. Старайтесь размещать системные блоки так, чтобы горячий "выхлоп" одного компьютера не попадал на воздухозаборник другого.

Пользователям ноутбуков можно помочь с помощью подставок для охлаждения корпуса ноутбука. Эти подставки оснащены одним или двумя вентиляторами, охлаждающими корпус установленного сверху ноутбука (рис. 45.1). КПД подставки довольно низкий, поскольку охлаждается корпус, но не сам процессор. Однако такая подставка лучше, чем ничего — она ведь также охлаждает и жесткий диск ноутбука. Толк от такой подставки будет, если ноутбук предварительно очищен от пыли. Иначе воздух от вентиляторов просто не будет проникать в вентиляционные отверстия корпуса ноутбука.



Рис. 45.1. Подставка для охлаждения ноутбука

У таких подставок есть еще один недостаток — цена. Кондиционер начального уровня стоит примерно 4500 рублей. А вот одна такая подставка— от 1000 до 2000 рублей. Выходит, четыре подставки равны одному кондиционеру. Если руководство компании не потратилось на кондиционер, то сомневаюсь, что оно выделит деньги на подставки.

45.5. Стойки для оборудования

Для большего порядка в серверной рекомендуется применять серверные стойки или шкафы. Шкаф стоит дороже, но зато его можно закрыть на ключ, что предотвращает несанкционированный физический доступ к серверу, даже если кто-то случайно окажется в серверной комнате. Типичный серверный шкаф изображен на рис. 45.2.

Как видите, кроме самих серверов в шкаф можно поместить и сетевое оборудование, что очень удобно. На рис. 45.3 изображена серверная стойка.



Рис. 45.2. Серверный шкаф



Рис. 45.3. Серверная стойка

45.6. Влажность

Идеальная влажность для компьютерных систем — от 40 до 55%. Если влажность низкая, возникнут проблемы с электростатическими зарядами. Если же влажность слишком высокая, влага будет конденсироваться на платах, что вызовет окисление контактов и замыкание. Бороться с высокой влажностью можно с помощью кондиционеров с функцией осушения воздуха. А вот если влажность низкая, подойдут увлажнители воздуха (самый дешевый стоит около 2000 рублей).

Что касается статического электричества, то, прежде чем приступить к разборке компьютера, лучше всего полностью выключить его (из розетки в том числе) и воспользоваться антистатическим браслетом (рис. 45.4). Такой браслет надевается на запястье и подсоединяется к "заземлению" — к третьему штырьку в розетке, если таковой имеется…



Рис. 45.4. Антистатический браслет

45.7. Инструмент системного администратора

Любому системному администратору пригодятся следующие инструменты:

- набор плоских и крестообразных отверток — всегда нужен, ведь разбирать и собирать компьютеры придется часто;
- набор мелких ювелирных отверток — вы их будете использовать редко, но хорошо, если таковые будут под рукой, чтобы потом не пришлось в спешке их покупать;
- набор шестигранных ключей, набор ключей типа TORX (шестигранная звезда) — без таких ключей некоторые устройства (например, ноутбуки) разобрать не получится;
- пинцет — некоторые мелкие вещи, которые вы уроните в системный блок, намного удобнее доставать с помощью пинцета;
- фонарик — освещение не всегда отличное, поэтому фонарик тоже нужен, желательно купить светодиодный фонарик — он лучше светит;
- ножницы — на всякий случай;
- инструмент для обжима витой пары — думаю, зачем он нужен, говорить не нужно;
- запасные разъемы RJ-45 — стоят копейки, поэтому чем больше, тем лучше (в пределах разумного, конечно);
- инструмент для зачистки проводов — для зачистки проводов можно также использовать и инструмент для обжима витой пары, но лучше купить отдельный специальный инструмент;
- антистатический браслет — ранее мы говорили, зачем он нужен;
- кроссовер — может пригодиться для прямого (в обход коммутатора) соединения двух компьютеров;
- обжатый Ethernet-кабель — пригодится для подключения вашего ноутбука к коммутатору (для тестирования соединения), минимальная длина такого кабеля — 1 метр;
- цифровой мультиметр — объединяет в себе несколько измерительных приборов, как правило, это вольтметр, амперметр и омметр (рис. 45.5).



Рис. 45.5. Цифровой мультиметр

Вместо заключения

Вместо заключения приглашаю всех читателей на форум моего сайта **www.dkws.org.ua**. Если у вас есть какие-нибудь вопросы, не получается что-то настроить, вы можете смело обращаться на форум. Перед тем как задать вопрос на форуме, прочитайте справочную систему (команда `man`), файл примера (для многих конфигурационных файлов существуют файлы примеров, где тщательно прокомментирована каждая опция конфигурационного файла) и воспользуйтесь поиском по форуму. В большинстве случаев ответ на свой вопрос вы получите еще до того, как успеете задать его на форуме.

Также не забывайте об официальном руководстве и приложениях к этой книге — они обязательны для прочтения.



ПРИЛОЖЕНИЯ



Приложение 1

Двойная загрузка: Windows 7 и FreeBSD

В Интернете есть множество способов организации двойной загрузки Windows и FreeBSD. Как правило, все эти способы ориентированы на Windows XP, поскольку сводятся к редактированию файла загрузчика boot.ini. В современных версиях Windows (Vista и 7) конфигурация загрузчика изменяется иначе, поэтому в этом приложении мы рассмотрим совместную загрузку именно Windows 7 и FreeBSD. Нужно же быть впереди планеты всей?

Существует два способа организации двойной загрузки FreeBSD и Windows 7. Точнее, способов может и больше, но лично я опробовал только два способа, которые и будут рассмотрены в этом приложении.

Первый способ подразумевает, что у вас есть компьютер, на котором еще не установлена операционная система. Сначала нужно установить FreeBSD. При установке выберите или стандартный загрузчик, или BootMgr — имейте в виду, что BootMgr обеспечит наличие меню при загрузке FreeBSD. После установки скопируйте файл /boot/boot1 на флешку.

Перезагрузите компьютер и установите Windows 7. Скопируйте файл boot1 в корневой каталог диска C: под именем FreeBSD.mbr.

Запустите командную строку (`cmd.exe`) с правами администратора и введите команду:

```
bcdedit /create /d "FreeBSD 8" /application bootsector
```

В ответ вы получите что-то вроде этого:

The entry {01234567-ABCD-ABCD-ABCD-0123456789AB} was successfully created.

Значение в фигурных скобках нужно записать (или скопировать в текстовый редактор). Далее введите следующие команды, в которых замените фрагмент `{значение}` на только что полученное:

```
bcdedit /set {значение} device boot
bcdedit /set {значение} path \FreeBSD.mbr
bcdedit /set {значение} device partition=c:
bcdedit /displayorder {значение} /addlast
```

Теперь рассмотрим **второй способ**, который заключается в использовании программы EasyBCD, которую можно скачать по адресу:
http://neosmart.net/dl.php?id=1.

В этом случае будем считать, что Windows 7 уже была установлена первой. Установите FreeBSD, но при установке вообще не устанавливайте загрузчик. Снова загрузитесь с загрузочного диска FreeBSD, запустите программу fdisk и сделайте раздел с Windows активным (загрузочным).

Затем перезагрузитесь в Windows, запустите программу EasyBCD и добавьте FreeBSD в меню загрузчика (это можно сделать на вкладке **Linux** — да, это не опечатка, вкладка называется именно так). Сохраните изменения и перезагрузитесь.

Приложение 2



Настройка загрузчика GRUB: Linux и FreeBSD

В этом приложении мы рассмотрим, как обеспечить загрузку FreeBSD с помощью загрузчика GRUB/GRUB2. Начнем с обычного GRUB. Чтобы загрузить FreeBSD достаточно в конфигурационный файл загрузчика /boot/grub/menu.lst добавить следующие строки:

```
# FreeBSD
title  FreeBSD
root   (hd0,1)
kernel (hd0,1)/boot/loader
```

После перезагрузки в меню загрузчика появится новый пункт меню — выбираете его и загружаете FreeBSD. Все достаточно просто.

Но вы должны знать, что ранние версии GRUB (до 0.94, то есть до конца 2005 года) поддерживали только файловую систему UFS, а в FreeBSD, начиная с 5-ой версии, по умолчанию используется файловая система UFS2. С современными версиями GRUB, которые, например, установлены в Mandriva 2010 или Ubuntu 9.04 проблем быть не должно.

Более перспективный загрузчик, используемый в современных дистрибутивах Linux, — GRUB2. С файловой системой UFS2 он работает отлично, чего не скажешь об ZFS. Впрочем, имеется патч, добавляющий поддержку ZFS, но от использования этой файловой системы лучше пока отказаться. Она экспериментальная, да и GRUB2 еще находится на стадии тестирования, поэтому сочетание GRUB2 + ZFS можно порекомендовать только любителям экстрима. Всем остальным пользователям беспокоиться нечего. Откройте файл /etc/grub.d/40_custom и добавьте в него следующие строки:

```
menuentry "FreeBSD 8.0" {
        set root=(hd0,1,a)
        chainloader +1
}
```

Здесь имеется в виду, что FreeBSD установлена на первом винчестере (hd0), то есть на /dev/sda, на первом разделе (/dev/sda1), имя BSD-раздела — a.

После изменения файла /etc/grub.d/40_custom выполните команду `sudo update-grub`.

Приложение 3



Проблемы с USB-накопителями в FreeBSD 8.0

Это приложение относится только к FreeBSD версии 8.0-RELEASE. В версии 8.1 проблема исправлена.

Проблема замечена на материнских платах с чипсетом nForce и проявлялась в остановках при копировании данных с флешки или USB-винчестера на обычный жесткий диск и обратно. На других чипсетах все работает нормально.

Для устранения проблемы откройте файл /usr/src/sys/dev/usb/controller/ehci.c и найдите в нем следующие строки:

```
static void
ehci_device_bulk_start(struct usb_xfer *xfer)
{
    ehci_softc_t *sc = EHCI_BUS2SC(xfer->xroot->bus);
    uint32_t temp;

    /* setup TD's and QH */
    ehci_setup_standard_chain(xfer, &sc->sc_async_p_last);

    /* put transfer on interrupt queue */
    ehci_transfer_intr_enqueue(xfer);

    /* XXX Performance quirk: Some Host Controllers have a too low
    * interrupt rate. Issue an IAAD to stimulate the Host
    * Controller after queueing the BULK transfer.
    */
    temp = EOREAD4(sc, EHCI_USBCMD);
    if (!(temp & EHCI_CMD_IAAD))
    EOWRITE4(sc, EHCI_USBCMD, temp | EHCI_CMD_IAAD);
}
```

Строки, выделенные полужирным шрифтом, следует удалить. Затем сохраните файл и перекомпилируйте ядро.



Приложение 4

Основные сетевые устройства

П4.1. Активное и пассивное сетевое оборудование

Для построения компьютерной сети, то есть для организации передачи информации между компьютерами, используется сетевое оборудование. Сетевое оборудование бывает активным и пассивным. *Активным* называется оборудование, обладающее неким "интеллектом" — например, коммутатор (switch), маршрутизатор (router). *Пассивное* сетевое оборудование "интеллектом" не наделено. К пассивному оборудованию относят кабели (например, коаксиальный или витая пара), розетки (RJ45, RG58 и др.), повторитель (repeater), концентратор (hub) и т. д.

Стоп! Если вы хоть немного знакомы с Ethernet-сетями, то можете запутаться. Ведь концентратор, как и коммутатор, можно использовать в качестве центрального сетевого устройства в Ethernet-сети, почему тогда концентратор — это пассивное устройство, а коммутатор — активное? Дело в том, что концентратор не проявляет никакой интеллектуальной деятельности — он просто получает сигналы и копирует (повторяет) их на все свои порты, равно как и повторитель. Повторитель получает сигнал, усиливает его и повторяет на другой порт. Повторители обычно используются для увеличения дальности передаваемого сигнала. Коммутатор же "знает", к какому порту подключен какой компьютер, поэтому передает полученный сигнал не на все порты, а только на определенный порт, к которому подключен компьютер-назначение.

Различного сетевого оборудования очень много. Мы не будем пытаться объять необъятное, поэтому в этой книге рассмотрим только оборудование, необходимое для построения проводных Ethernet-сетей и беспроводных сетейWi-Fi.

П4.2. Оборудование, необходимое для построения Ethernet-сети

Для организации современной Ethernet-сети (имеются в виду спецификации Fast Ethernet и Gigabit Ethernet) необходим всего один коммутатор (switch). Конечно, если сеть большая, то понадобится несколько коммутаторов, общее количество портов которых сможет обеспечить подключение всех узлов сети. На рис. П4.1 изображен так называемый *промышленный* коммутатор от Linksys.

Дизайн корпуса промышленного коммутатора обычно не очень эффектен, но сделано это умышленно — чтобы коммутатор можно было поместить в стойку сетевого оборудования. Ведь в больших корпоративных сетях обычно несколько коммутаторов, помещаемых в специальную стойку (или в специальный шкаф сетевого оборудования, который можно закрыть и тем самым ограничить физический доступ к нему). На рис. П4.2 изображена типичная стойка с коммутаторами.

А на рис. П4.3 изображен шкаф с коммутаторами. Такой шкаф может быть большего размера и содержать другое оборудование (например, серверы сети), но главное отличие шкафа от стойки — наличие двери, которая ограничивает доступ к сетевому оборудованию.



Рис. П4.1. 16-портовый коммутатор от Linksys

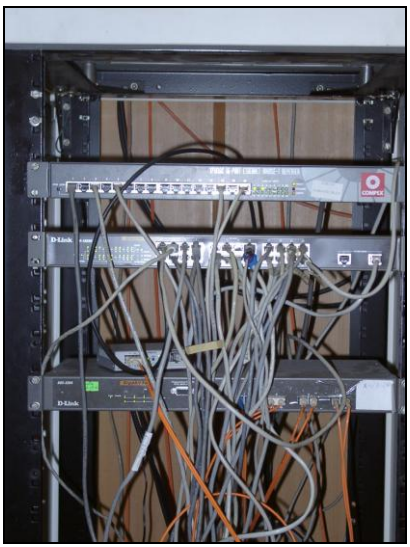


Рис. П4.2. Стойка с коммутаторами



Рис. П4.3. Шкаф с сетевым оборудованием



Рис. П4.4. 8-портовый гигабитный коммутатор от D-Link

Если вы хотите построить небольшую домашнюю или офисную сеть, то можете выбрать коммутатор с более интересным дизайном, который лучше впишется в ваш интерьер. На рис. П4.4 изображен 8-портовый гигабитный коммутатор от D-Link. Вид у него более "дружелюбный", но в стойку его уже не поместишь, хотя при организации домашней сети никакой стойки у вас и не будет.

Давайте теперь уточним, почему в современных сетях не стоит использовать концентраторы (hub). Представим, что у нас есть сеть на четыре компьютера. Назовем их А, Б, В и Г. Пусть компьютер А отправляет данные компьютеру Г. Концентратор отправит полученный от компьютера А сигнал на все свои порты — то есть сигнал, отправленный компьютером А, получат все компьютеры сети. Затем каждый компьютер анализирует заголовки пакета, в которых указан компьютер-получатель. Если адрес компьютера совпадает с адресом получателя, компьютер принимает пакет, в противном случае — игнорирует его. Таким образом, использование концентратора приводит к "брожению" по сети паразитного трафика. По сути, концентратор — это обычный многопортовый повторитель (усилитель) сигналов. И чем больше сеть, тем медленнее она работает в случае использования концентратора, поскольку "брожение" паразитного трафика имеет лавинообразный характер. Вы только представьте, что в сети не четыре компьютера, а несколько десятков… Поэтому в больших сетях концентраторы существенно снижают производительность сети.

Коммутатор же, в отличие от концентратора, строит специальную таблицу соответствия, позволяющую однозначно узнать, к какому порту какой компьютер подключен (см., например, табл. П4.1).

Таблица П4.1. *Таблица соответствия портов коммутатора и адресов компьютеров*

Номер порта	Адрес компьютера
1	Б
2	А
3	Г
4	В

Когда компьютер А, подключенный ко второму порту коммутатора, отправляет пакет компьютеру Г, коммутатор знает, что компьютер Г подключен к третьему порту, и отправляет пакет только на третий порт. При этом снижается нагрузка на сеть, потому что компьютеры не получают "лишних" пакетов.

Кроме того, поскольку концентратор отправляет данные каждому компьютеру сети, становится очень простым перехват данных. Существуют специальные программы, переводящие сетевой адаптер в режим мониторинга, в котором он осуществляет принятие всех данных, даже тех, которые не адресованы этому компьютеру. Поэтому, если в сети используется концентратор, все передаваемые данные становятся общим достоянием — их может перехватить любой компьютер, подключенный к концентратору.

Итак, использование коммутатора позволяет повысить и производительность сети, и ее безопасность. Ранее сети в основном строились на базе концентраторов, поскольку их стоимость была существенно ниже стоимости коммутаторов. Со снижением цен на коммутаторы концентраторы практически исчезли с магазинных полок. Однако в некоторых старых сетях они еще используются. Если вам придется обслуживать такую сеть, первым делом замените концентратор на коммутатор — вы сразу почувствуете разницу.

Какой коммутатор применить: Fast Ethernet (100Base-T) или Gigabit Ethernet (1000Base-T)? В первом случае максимальная (теоретическая) скорость передачи данных составляет 100 Мбит/с, во втором случае — 1000 Мбит/с. Коммутаторы Gigabit Ethernet стоят немного дороже (цены приводить не буду, поскольку через год они станут еще доступнее, а через два — о Fast Ethernet забудут, как в свое время забыли о коаксиале и концентраторах).

Учитывая, что сеть строится не на день и не на два, лучше выбрать Gigabit Ethernet. С точки зрения монтажа сети ничего не изменится — даже если вы сейчас установите коммутатор Fast Ethernet, то завтра без проблем сможете заменить его на Gigabit Ethernet. Но нужно помнить следующее: чтобы сеть работала в режиме 1000Base-T, необходимо, чтобы 1000Base-T поддерживали сетевые адаптеры компьютеров. Практически на всех современных материнских платах встроенные сетевые адаптеры уже поддерживают 1000Base-T, но если в вашей сети есть компьютеры, которым 2–3 года, скорее всего, вам придется докупать для них сетевые адаптеры с поддержкой 1000Base-T.

Идем дальше — количество портов. Обычно в продаже есть коммутаторы на 5, 8, 16, 24 порта. Промышленные коммутаторы могут иметь большее число портов, например 32 или 48. Может быть, в скором времени появятся коммутаторы с большим числом узлов, но я сомневаюсь. Поскольку обычно один коммутатор обслуживает одну подсеть, я не думаю, что в одной подсети будет больше 48 компьютеров. А если это случится, такую подсеть желательно (из соображений локализации трафика) разделить на несколько подсетей с меньшим числом компьютеров.

Так что для домашней сети покупайте коммутатор, способный подключить все имеющиеся дома компьютеры, — большой запас портов вам вряд ли понадобится. Обычно в домашней сети 2–4 компьютера. В этом случае вам будет достаточно 5-портового коммутатора — 5-й порт пригодится для подключения этого коммутатора к другому коммутатору сети. В коммутаторах с большим числом портов для подключения к другому коммутатору обычно используется один из имеющихся портов (например, порт 1). Промышленные коммутаторы иногда имеют так называемый *магистральный* порт. Например, 16 портов, работающих в режиме 100Base-T, и один порт, работающий в режиме 1000Base-T, — для подключения к магистрали сети, работающей со скоростью 1000 Мбит/с. Иногда вместо порта 1000Base-T оборудуется оптоволоконный порт, например, 100Base-FB. В этом случае скорость магистрали такая же, как и скорость сети, но расстояние передачи сигнала намного выше (более 2 км), что позволяет использовать оптоволоконный кабель для соединения сетей двух (или более) зданий в одну большую сеть.

В случае с офисной сетью количество портов коммутатора должно в два раза превышать количество компьютеров сети. Например, если в вашей сети четыре компьютера, то нужен 8-портовый коммутатор. Дополнительные четыре порта могут понадобиться, если придется подключить дополнительные компьютеры, например, ноутбуки ваших клиентов, если у вас пока еще нет для них точки доступа Wi-Fi.

По большому счету, для организации сети больше ничего и не нужно (разумеется, кроме кабеля и коннекторов RJ45, но это уже детали, о которых рассказано в *главе 22*).

П4.3. Оборудование, необходимое для построения сети Wi-Fi

Как и в случае с Ethernet-сетью, нам понадобятся сетевые адаптеры и центральное устройство сети. Только сетевые адаптеры нужны не обычные, а беспроводные. А роль центрального устройства сети будет играть *точка доступа* (access point).

Все современные модели ноутбуков по умолчанию оснащены адаптером Wi-Fi, а вот стационарные (настольные) компьютеры придется дооснастить беспроводными сетевыми адаптерами. Проще всего купить беспроводной адаптер, подключающийся к компьютеру по USB. Есть также адаптеры, выполненные в виде PCI-карты, устанавливаемой в свободный PCI-слот компьютера. Такие адаптеры используются редко, поскольку их установка требует вскрытия корпуса компьютера, что несколько неудобно (особенно, если компьютер еще на гарантии — тогда придется нести его в сервисный центр, а что делать, если таких компьютеров много?).

USB-адаптеры могут быть выполнены в разных корпусах. На рис. П4.5 изображен небольшой беспроводной адаптер, напоминающий по своим размерам флешку. У такого адаптера антенна встроенная, поэтому его можно использовать только, если компьютер находится в зоне уверенного приема. Если же компьютер установлен ближе к "мертвой" зоне, лучше выбрать адаптер, выполненный в виде отдельного устройства (рис. П4.6). Такой адаптер обычно имеет небольшой размер и подключается к компьютеру USB-кабелем (питание адаптер получает тоже по USB). Преимущество этого адаптера заключается в следующем — его можно легко передвинуть в пределах длины USB-кабеля, чтобы попасть в зону уверенного приема сети. Ноутбук можно легко переместить в эту зону — просто взяли и перенесли. Со стационарным компьютером такого не сделаешь — у каждого стационарного компьютера есть свое место. А что делать, если в том месте, где установлен компьютер, не обеспечивается уверенный прием беспроводных сигналов? Не переносить же компьютер? В этой ситуации поможет адаптер, изображенный на рис. П4.6. Иногда перемещение адаптера всего на несколько сантиметров дает весьма ощутимые результаты. Да и антенна у такого адаптера обладает большей чувствительностью, чем встроенная антенна адаптера, изображенного на рис. П4.5. К тому же к подобным адаптерам (с внешней антенной) обычно можно подключить дополнительную антенну с еще большей чувствительностью. Обо всем этом мы поговорим, когда будем строить свою собственную беспроводную сеть. А сейчас перейдем лучше к точке доступа.

Примечание

При выборе адаптера Wi-Fi учитывайте наличие драйверов — особенно, если вы планируете использовать его в FreeBSD. Чтобы не получилось так, что FreeBSD не поддержит купленный Wi-Fi-адаптер (а это может произойти!).

Точка доступа (рис. П4.7) выполняет в беспроводной сети роль центрального устройства. Казалось было, все здесь просто: устанавливаем Wi-Fi-адаптеры, подсоединяем точку доступа, и беспроводная сеть готова — беспроводные клиенты могут обмениваться данными. Однако, если вы планируете купить точку доступа прямо сейчас, не следует покупать первую попавшуюся. Сначала желательно определиться, какие функции точки доступа вам нужны, затем "вычислить" модели точек доступа, обеспечивающие необходимые вам функции, и просмотреть в Интернете отзывы об этих моделях. Только так можно выбрать лучшую точку доступа.

Точка доступа может предоставлять дополнительные функции — например, функции *маршрутизатора*. Предположим, у вас дома есть несколько ноутбуков. К одному ноутбуку подключен ADSL-модем. Как организовать общий доступ к Интернету? Покупается точка доступа, к которой этот ADSL-модем и подключается. Ноутбуки (беспроводные клиенты) будут подключаться к Интернету по Wi-Fi, а точка доступа выступит в роли маршрутизатора.



Рис. П4.5. USB Wi-Fi-адаптер со встроенной антенной



Рис. П4.6. USB Wi-Fi-адаптер с внешней антенной



Рис. П4.7. Точка доступа от D-Link с тремя антеннами

П4.4. Дополнительные сетевые устройства

Представим, что у нас есть два (или более) обычных (настольных) компьютера и одно ADSL-соединение. И нужно обеспечить общий доступ к Интернету. Это можно сделать средствами операционной системы. Тогда в один компьютер надо будет установить дополнительный сетевой адаптер. Первый сетевой адаптер будет использоваться для подключения к Интернету, а второй — для подключения к локальной сети (для связи с остальными компьютерами сети). Компьютер с двумя сетевыми адаптерами для остальных компьютеров сети будет выполнять роль *шлюза* (gateway). Преимущество такого решения — дешевизна: ведь мы обеспечили общий доступ к Интернету практически без дополнительных устройств. Недостаток заключается в том, что компьютер-шлюз должен быть постоянно включен, иначе остальные компьютеры не смогут подключиться к Интернету.

Решить эту проблему можно, купив отдельное устройство, называемое *маршрутизатором* (при рассмотрении выбора точки доступа мы это устройство уже упоминали). Маршрутизатор обеспечивает передачу пакетов по заданному маршруту. В нашем случае — от локальных компьютеров к интернет-провайдеру. Таким образом, все компьютеры сети будут подключаться к центральному коммутатору, а он, в свою очередь, — к маршрутизатору. Также к маршрутизатору будет подключен и ADSL-модем.

Маршрутизаторы бывают разные. Некоторые могут выполнять роль коммутатора. Купив такой маршрутизатор, вы сократите количество активного сетевого оборудования (а значит, сэкономите деньги) до двух единиц — маршрутизатора и ADSL-модема. Если же у вас в сети компьютеров немного (2–4), можно подыскать ADSL-модем с функциями маршрутизатора. В этом случае у вас будет всего одна "коробочка" — все компьютеры сети будут подключены к этому устройству, которое, в свою очередь, будет подключено к телефонной сети. Этим вы сэкономите еще больше средств. Поэтому очень важно перед построением сети спланировать сей процесс. Хорошее планирование не только позволяет сэкономить деньги, но и время, впоследствии потраченное на дальнейшую модернизацию сети.

А теперь представим, что в нашей сети есть два (или больше, количество — не принципиально) стационарных компьютера и несколько ноутбуков. Ноутбуки было бы хорошо подключать по Wi-Fi. Стационарные компьютеры принято подключать по Ethernet (хотя бы потому, что не хочется покупать для них беспроводные адаптеры). Так вот, можно купить устройство, которое одновременно является ADSL-модемом, беспроводной точкой доступа и коммутатором. Одним из таких устройств является DSL-2640U от D-Link (далее мы рассмотрим процесс настройки этого устройства). Это устройство (рис. П4.8) позволяет объединить в сеть несколько беспроводных клиентов (это наши ноутбуки) и четыре проводных клиента. Все клиенты (как проводные, так и беспроводные) автоматически настраиваются на доступ к Интернету по совместно используемому ADSL-каналу. Кроме того, это устройство обладает встроенным брандмауэром, что позволяет защитить вашу сеть от вторжения извне.



Рис. П4.8. ADSL-модем, маршрутизатор, коммутатор и беспроводная точка доступа D-Link DSL-2640U

Простота настройки сети с помощью такого устройства просто поражает. Все, что вам нужно — это включить устройство, подключить к нему клиентов, запустить программу настройки (как это сделать, написано в руководстве по устройству) и установить базовые параметры сети, а именно: имя пользователя и пароль для ADSL-соединения, идентификатор беспроводной сети (SSID) и выбрать тип шифрования беспроводных соединений. Вот и все — сеть будет работать. Клиентов можно вовсе не настраивать — они будут автоматически настроены по протоколу DHCP (Dynamic Host Configuration Protocol, протокол динамической настройки узла).

Впрочем, у всех комбинированных устройств есть один недостаток — плохая масштабируемость. Если ваша сеть будет расти, добавить новых клиентов в нее будет сложно, а в некоторых случаях вовсе невозможно. Тогда придется покупать отдельные устройства. Например, коммутатор, к которому будут подключаться до 48 проводных клиентов, и точку доступа для подключения беспроводных клиентов. В свою очередь, точка доступа и коммутатор будут подключаться к ADSL-маршрутизатору. Хотя в сложных случаях целесообразнее использовать программный (не аппаратный) маршрутизатор — компьютер под управлением UNIX/Linux. Такой компьютер можно использовать в роли маршрутизатора и на нем запустить брандмауэр, DNS-, WWW-, FTP- и почтовый серверы.

Предметный указатель

1

1000BASE-X 278
10GBASE 278

3

3Com 275
3DES 401
3G/EDGE 434

A

Access point 513
ACL 359
ACPI 15
ADSL 278
ADSL-сплиттер 318
Almquist shell 125
Alohanet 274
Apache 406, 443
Apple 275
AT 274
AT&T 122
ATM 277

B

bash 121, 123, 125
Blowfish 208, 401

C

Callback 426
Centronics 273
Collapsed-backbone 277
csh 122
CSMA 275
CSMA/CD 275

D

DES 208
DHCP 328, 516
DHCP-сервер 17, 29, 78, 332, 407
DIX 275
DNS 286, 290, 336
DNS-сервер 336, 444, 445
  вторичный 346
  кэширующий 339
  первичный 343
DOS 174
DSA-ключ 405

E, F

Ethernet 275
Fast Ethernet 512
FDDI 277
fdisk 18, 19, 20, 21
Firewall 348
FireWire 273
Frame Relay 274, 277
FreeBSD 122
FTP 286, 369

G

Gateway 515
GID 208
Gigabit Ethernet 512
GRUB/GRUB2 507

H

Hayes AT 274
Hosts 290
HTTP 286
Hub 509

I

IDEA 401
IEEE 275
IEEE 1394 273
IEEE 802.1D 276
IEEE 802.3 275
IEEE 802.3a 275
IEEE 802.3i 276
IEEE 802.3u 277
IMAP 286
IMAP/POP3-сервер 386
IP 284
IPng 287
IPv6 287, 313
IP-адрес 286
ISDN 276
ISO-образ FreeBSD 192
ISO-образ LiveCD 59

K, L

ks 123
LAN 278
LAN Manager 276
LiveCD 59, 60, 61, 62, 69, 73, 74, 75
LiveUSB 76

M

MAC-адрес 284, 331, 332
MAN 278
MD5 208
MTA 386
MySQL 378, 379, 381, 397
MySQL-клиент 452
MySQL-сервер 412, 452

N

NAT 287, 348
NCSA HTTPd 1.3 406
NFS 382
NIC 287
Novell Netware 276

O

OpenBSD 39, 40, 41, 44, 45, 47
OSI 283

P

PAE 268
pdksh 123
Personal Computer 274
PID 220
POP 286
POSIX 122
PPTP 427
ProFTPD 444

Q, R

QoS 277
RAID 257
RAS 426
Remote Administrator 201
Repeater 509
Router 509
RS-232C 273
RSA 401
RSA-ключ 405

S

Samba 322, 446
SGID 184
sh 122
Slice 19
SMS 475
SMTP 286, 386
SNA 274
SOA 345
Squid 356
SSH-сервер 402
SSID 516
SSL 285
SUID 184
Switch 509

T

T1 275
TCP 285
TCP/IP 285
tcsh 125
Telnet 401
TENEX 125
TLD 336
Token Ring 275
TSIG 444

U, V

UID 208
UNIX 89, 276
VPN-клиент 438

W

WAN 278
Web-сервер 313, 406, 408, 411, 418, 423

X

X Window 98
X.25 274
X.Org 98, 99, 101, 102, 111

Z

zsh 124

А

Автодополнение 85
Анализатор протоколов 363
Антивирус ClamAV 471
Аутентификация 459

Б

Брандмауэр 348, 433, 434, 436

В

Внешний
  жесткий диск 57, 58, 59
Выражение 144

Г

Графическая среда 98, 99
Графический интерфейс 98, 103

Д

Дамп системы 73, 75
Двойная загрузка 505
Демон HAL 111, 112, 113
Директива:
  default-leased-time 331
  DefaultRoot 376
  Directory 419
  Files 421
  Limit 420
  MaxClients 376
  max-leased-time 331
  ServerName 417
Диск гибкий 186
Домен 343

З, К

Зона 343
Кадр 284
Каталог:
  /etc 58
  /etc/skel 131, 209
  /home 59
  домашний 180
  признак каталога 182
  родительский 180
  текущий 179
Квотирование 210
Кодировка KOI8-R 96, 110
Кодировка UTF-8 96, 110
Коллекция портов 27
Команда:
  adduser 204, 207
  alias 132
  cat 177
  cd 179
  chattr 185
  chmod 182, 183
  chmod +x 135
  chown 184
  clear 154
  cp 177
  date 154
  df 160
  diff 156
  exit 154, 204
  free 161, 162
  fsck 191
  ftp 157
  grep 156
  head 157
  ifconfig 133

(окончание рубрики см. на стр. 520)

Команда *(окончание)*:
kill 220
killall 222
less 156, 178
links 159
ln 181, 182
locate 178
logout 86
ls 179
lynx 159
man 152
md5sum 160
mkdir 179
more 156
mount 185
mv 177
nice 226
nslookup 342
passwd 154, 207
pptp-command 436
ps 220
pw 446
rm 178, 179
rmdir 179
rmuser 207
rndc-confgen 343
route 311
scp 71
setenv 132
shutdown 86
smbpasswd 323, 446
ssh 402
su 204
sudo 202
tac 177
tail 157
tar 70
top 223
touch 177
umount 186
uname 152, 153
uptime 154
users 155
wc 157
which 178
who 155, 201
Коммутатор 509, 511
Коммутация 282
Компиляция ядра 264
Консоль 83, 84, 85, 86, 88, 92, 93, 94, 96
Конфигуратор sysinstall 34, 87, 308
Концентратор 509, 511
Корневая файловая система 165, 167, 170, 171, 172, 175, 181, 186, 192

М

Маршрутизатор 348, 509
Маска сети 288
Массивы 137
Мир 269
Мониторинг:
сети 456
трафика 449, 452

О

Оборудование:
активное 509
пассивное 509
Образы FreeBSD 13
Оператор 144
case 139
if 138

П

Пакет 228, 237, 238, 240
pptp-client 436
pptp-linux 436
зависимости 238
конфликты 238
Пароль root 17, 36, 42
Переменные 135
окружения 136, 142
специальные 136
Перехват ICQ-трафика 463
Планировщик crond 217
Повторитель 509
Полнодуплексный режим 311
Полудуплексный режим 311
Пользователь root 202
Порт 228, 232, 235, 236, 237
Привод CD-RW 56
Привод DVD-RW 57, 72
Программа:
mc 71
rndc 342
Clonezilla 59
Программный RAID-массив 166, 259

Прозрачный прокси-сервер 364
Прокси-сервер 356, 470
Протокол 285
  SSH 401
  TCP/IP 327
Прототипы 212

Р

Редактор vi 89
Редактор разделов 23, 24
Резервная копия: выбор носителя 55
РУС-BSD 51

С

Сервер DNS 444
Сервер TFTP 79
Сертификат 429
Сетевой адаптер 305, 306, 307, 311
Сетевой интерфейс 305, 310, 350
Сетевой сканер 466
Сеть:
  клиент/сервер 282
  одноранговая 282
  топология 279
Система:
  доменных имен 336
  печати CUPS 241, 253, 326
  печати lpr 241
Слайс 19, 20, 21, 170
Сниффер 463, 464
Соединение:
  DSL 318
Стример 56
Сценарий 122, 134, 135, 136, 137, 138, 140, 141, 142, 147

Т

Технология gdbe 479
Топология:
  дерево 280
  звезда 280
  кольцевая 279
  линейная 279
  полносвязная 280
  шина 280
  ячеистая 280
Точка доступа 513

Ф

Файл:
  .bash_history 126
  /etc/crontab 217
  /etc/csh.cshrc 131
  /etc/csh.login 131
  /etc/csh.logout 131
  /etc/dhcpd.conf 329
  /etc/fstab 191, 211
  /etc/passwd 208
  /etc/profile 126
  /etc/proftpd/proftpd.conf 373
  /etc/resolv.conf 342
  /etc/shadow 208
  /etc/shells 121, 444
  /etc/sshd_config 403
  ~/.bash_logout 126
  ~/.bash_profile 126
  ~/.bashrc 126
  ~/.history 131
  dnssec-keygen 444
  etc/squid/squid.conf 356
  smb.conf 326
  права доступа 182
  устройства 186
Файловая система:
  ext2 165
  ext3 165, 166, 167, 190
  geli 479
  ReiserFS 165, 166, 167, 172, 189
  UFS 165, 166, 167, 170, 172, 186, 193
  UFS2 165, 166, 167, 169
  ZFS 165, 166, 167, 168, 169, 189, 190
Файловые системы 165, 166, 167, 177, 187, 190, 191, 192
Форвард-сервер 341

Ц, Ч

Цикл:
  for 137
  while 138
Черный список 359

Ш, Э

Шифрование диска 480
Шлюз 320, 348, 515
ЭВМ 273